# Entwicklung einer modernisierten Fernsteuerung für die Kölner Quelle polarisierter Ionen LASCO

Diplomarbeit
von
Frank Janus

Institut für Kernphysik
Universität zu Köln

- März 1996 -

## Zusammenfassung

Ziel der vorliegenden Diplomarbeit war die Entwicklung einer modernisierten, rechnergestützten Fernsteuerung für die Kölner Quelle polarisierter Ionen LASCO. Kern der Steuerung ist ein PC unter dem UNIX-kompatiblen Betriebssystem LINUX, der die Daten zur Quellensteuerung über einen seriellen CAMAC-Highway zu den Geräten auf der Quelle überträgt.

Im Rahmen dieser Arbeit wurde der serielle Highway, der zur Trennung von Erd- und Quellenpotential über eine serielle Optokopplerstrecke verfügt, aufgebaut und getestet. Auf dem PC wurde das Betriebssystem LINUX installiert und der PC an das Institutsnetz angebunden. Für die Anbindung an CAMAC wurden eigene Treiberroutinen entwickelt. Das Kontrollpult der bisherigen Steuerung wurde in die neue Steuerung integriert. Ein umfangreiches Steuerprogramm wurde entwickelt, welches alternative Bedienungsmöglichkeiten über Terminal oder Kontrollpult vorsieht, Protokollierung und Speicherung der Quellendaten ermöglicht, Möglichkeiten zur Fehlererkennung und Überwachung der Quelle bietet, die automatische Übernahme der auf der Quelle intern eingestellten Werte zuläßt und Steuerungsbefehle von anderen Programmen empfangen kann. Eine Shell für die Interpretersprache Tcl/Tk wurde entwickelt, die mit dem Quellenprogramm kommunizieren kann und mit deren Hilfe schnell und einfach grafische Benutzeroberflächen unter X-Windows für das Steuerprogramm erstellt werden können. Eine grafische Oberfläche wurde entwickelt. Die neue Steuerung wurde während zwei Strahlzeiten erfolgreich getestet.

Die neue Steuerung bietet gegenüber dem alten System den Vorteil einer erhöhten Betriebssicherheit und zahlreiche zusätzliche Möglichkeiten. Durch Anbindung an das Institutsnetz bzw. Internet ist eine Überwachung der Quelle von beliebigen Orten aus möglich. Durch Integration der Bedienungselemente der alten Steuerung entfallen für den Benutzer die Nachteile der Umstellung auf ein neues System.

# Inhaltsverzeichnis


**I Einleitung** **7**

**1 Motivation** **9**
1.1 Die Quelle polarisierter Ionen LASCO . . . . . 9
1.2 Aufbau der bisherigen Steuerung . . . . . 11
1.3 Modernisierung der Steuerung . . . . . 12

**2 Ausführung der neuen Steuerung** **15**
2.1 Hardware . . . . . 15
2.2 Software . . . . . 17

**II Hardware** **21**

**3 Überblick über die Hardware** **23**
3.1 Problemstellungen . . . . . 23
3.2 Problemlösungen . . . . . 23

**4 Aufbau des Serial Highway** **25**
4.1 Das CAMAC-System . . . . . 25
4.2 Rechner . . . . . 27
4.3 Der Serial Highway . . . . . 27
4.3.1 Technischer Aufbau und Konfiguration . . . . . 28
4.3.2 Software-Ansteuerung . . . . . 33
4.4 Quellencrate . . . . . 34

**5 Umbau des Kontrollpultes** **35**
5.1 Ansteuerung der Treiberplatinen . . . . . 35
5.1.1 Analog-Anzeige . . . . . 35
5.1.2 Digital-Anzeige (LEDs) . . . . . 38
5.1.3 Digital-Schalter . . . . . 38
5.1.4 Analog-Potentiometer . . . . . 38
5.1.5 Extern/Intern-LEDs . . . . . 38
5.2 Anbindung an das CAMAC-System . . . . . 38

**III Software** **41**

**6 Überblick über die Software** **43**
6.1 Problemstellungen . . . . . 43
6.2 Problemlösungen . . . . . 44

**7 Verwendete Techniken und Verfahren** **46**
7.1 Das Betriebssystem LINUX . . . . . 46
7.2 LINUX-Gerätetreiber für die CAMAC-Karte . . . . . 47
7.2.1 Programmierung der CAMAC-Karte . . . . . 47
7.2.2 Ansprechen der IO-Ports unter LINUX . . . . . 48
7.3 Tcl/Tk . . . . . 48
7.3.1 Konzept . . . . . 49
7.3.2 Die Schnittstelle zu C . . . . . 50
7.3.3 XF . . . . . 51
7.4 Signals . . . . . 51
7.4.1 UNIX Signal-Mechanismus . . . . . 52
7.4.2 Programmierung mit Signals . . . . . 52
7.5 Shared Memory . . . . . 54
7.6 Terminalprogrammierung unter UNIX . . . . . 54

**8 Aufbau des Steuerprogrammes** **57**
8.1 Programmablauf . . . . . 58
8.1.1 Initialisierung . . . . . 58
8.1.2 Hauptprogramm . . . . . 59
8.1.3 Signal-Routinen . . . . . 61
8.2 Datenübertragung . . . . . 65
8.2.1 Kommunikation mit der Quelle . . . . . 65
8.2.2 Kommunikation mit dem Kontrollpult . . . . . 67
8.2.3 Kommunikation mit dem Terminal . . . . . 68
8.2.4 Datenaustausch . . . . . 68
8.2.5 Fehlererkennung und Behandlung . . . . . 71
8.3 Das Client/Server-Konzept . . . . . 73
8.3.1 Funktion des Software-Interfaces . . . . . 73
8.3.2 Der Kommandozeileninterpreter . . . . . 74
8.3.3 Client-Programme . . . . . 75
8.4 Sonstige Einrichtungen . . . . . 76
8.4.1 CAMAC-Server . . . . . 76
8.4.2 Werteprotokollierung . . . . . 77
8.4.3 Werteübernahme . . . . . 77
8.4.4 Frei konfigurierbare Zusatzbits . . . . . 78

**IV Erfahrungen mit der Steuerung** **79**

**9 Test der Steuerung und Betriebserfahrung** **81**

**10 Beispiele für Anwendungen** **83**
10.1 Steuerung per Remote-Shell . . . . . 83
10.2 Tcl/Tk-Programmierung mit piqwish . . . . . 83
10.3 Konfiguration . . . . . 85
10.4 Alarmeinrichtungen . . . . . 86
10.5 Auswertung des Protokollfiles . . . . . 86

**11 Zusammenfassung** **88**
11.1 Ergebnisse . . . . . 88
11.2 Ausblick . . . . . 89

**V Anhang** **91**

**A Betriebsanleitung** **93**
A.1 Übersicht über die Steuerung . . . . . 93
A.2 Die Programme . . . . . 94
A.3 Starten des Systems . . . . . 96
A.4 Bedienung der Programme . . . . . 96
A.5 Bildschirmanzeige . . . . . 98
A.6 Bedienung über das Kontrollpult . . . . . 99
A.7 Bedienung über die Tastatur . . . . . 100
A.8 Bedienung über das X-Fenster . . . . . 101
A.9 Bedienung über Kommandozeilen . . . . . 102
A.10 Werteübernahme von der Quelle . . . . . 103
A.11 CAMAC-Kommandos . . . . . 105
A.12 Files . . . . . 105
A.13 Sonstiges . . . . . 106
A.14 Fehlersuche . . . . . 107

**B Technische Besonderheiten** **108**

**C Tastaturbelegung** **110**
C.1 Kommandozeilenoptionen piq . . . . . 110
C.2 analoge Parameter . . . . . 110
C.3 digitale Parameter . . . . . 110
C.4 Sonstiges . . . . . 111
C.5 Direktbelegung . . . . . 111

**D Befehlsübersicht** **112**
D.1 Analoge Parameter . . . . . 112
D.2 Digitale Parameter . . . . . 112
D.3 Setup-Befehle . . . . . 113
D.4 Quellenstatus erfragen . . . . . 114
D.5 Laden/Speichern des Quellenstatus . . . . . 114
D.6 Protokollfunktion . . . . . 115
D.7 Werteübernahme . . . . . 115
D.8 Sonstiges . . . . . 115

**E Tabellen** **116**

**Literaturverzeichnis** **120**

# Abbildungsverzeichnis


1.1 Die Kölner Quelle LASCO . . . . . 9
1.2 Das alte Fernsteuerungssystem . . . . . 11

2.1 Die neue Steuerung . . . . . 16
2.2 Programm piq . . . . . 17
2.3 Programm xpiq (Hauptmenü) . . . . . 18
2.4 Programm xpiq (CAMAC-Eingabe) . . . . . 19

4.1 CAMAC-System . . . . . 26
4.2 Signale des Serial Highway . . . . . 28
4.3 Serielle Übertragungsstrecke . . . . . 29
4.4 Signalweg U-Port-Adapter . . . . . 31

5.1 Kontrollpult-Signale . . . . . 36
5.2 Signale AAV . . . . . 37
5.3 Signale DAS . . . . . 37
5.4 Signale BT3 . . . . . 39
5.5 Signale AAV . . . . . 39
5.6 Signal-Timing . . . . . 40

7.1 Ansteuerung PC-Karte (Pseudo-Code) . . . . . 48
7.2 Ansteuerung PC-Karte (in C) . . . . . 49
7.3 Erweiterte Tcl/Tk-Shell in C . . . . . 50

8.1 Hierarchie des Quelltextes . . . . . 57
8.2 Ablaufplan des Hauptprogrammes . . . . . 60
8.3 Timer-Routine . . . . . 62
8.4 Keyboard-Routine . . . . . 64
8.5 Softwareinterface . . . . . 74
8.6 Listing piqcmd.c . . . . . 76

10.1 Listing: Steuerung per Remote-Shell . . . . . 83
10.2 Listing: Tcl-Skript für piqwish . . . . . 84
10.3 Ausschnitt aus einem piq_setup-File . . . . . 85
10.4 Beispiel für ein Alarmskript . . . . . 86
10.5 Konfiguration im File *piq_setup* zur Anzeige des Status der Turbo-Pumpen. . 86
10.6 AWK-Programm zur Werteextraktion . . . . . 87

A.1 Screenshot des Steuerprogrammes. . . . . 97

# Tabellenverzeichnis


1.1 Ferngesteuerte Netzgeräte . . . . . 10

4.1 CAMAC Dataway-Signale . . . . . 27
4.2 Konfiguration Master UPA . . . . . 30
4.3 Konfiguration Slave UPA . . . . . 32

8.1 Felder der Datenstruktur `piq_channel` . . . . . 68
8.2 Frequenzen der Übertragungszyklen . . . . . 71
8.3 Statusregister des SHD . . . . . 72

A.1 Direktbelegung der Tastatur . . . . . 101

E.1 Zusatzverkabelung Steuerpult . . . . . 116
E.2 Belegung IO440-Ausgaberegister . . . . . 117
E.3 Belegung IO440-Eingaberegister . . . . . 118
E.4 Belegung serielles Kabel . . . . . 118
E.5 Belegung AUX-Connector . . . . . 119
E.6 Belegung Turn-Round-Connector . . . . . 119
E.7 Bitmap Digitalschalter . . . . . 119
E.8 Bitmap LEDs . . . . . 119

# Teil I

# Einleitung

# Kapitel 1

# Motivation

## 1.1 Die Quelle polarisierter Ionen LASCO

Die Quelle polarisierter Ionen LASCO[1] dient der Erzeugung polarisierter Ionenstrahlen aus Wasserstoff- oder Deuteriumatomen unter Ausnutzung der Lambshift. Hierzu wird zunächst mittels einer Gasentladung ein Primärionenstrahl ($H^+, D^+$) aus durch ein Palladium-Ventil zugeführten $H_2$-Atomen erzeugt. Dies geschieht durch Stoßionisation mit schnell oszillierenden Elektronen, die durch eine eingestrahlte Hochfrequenz angeregt werden. Über der Entladungszone liegt das Magnetfeld eines Solenoids, das die Elektronen auf Spiralbahnen zwingt und so die Ionisationswahrscheinlichkeit erhöht. Das Palladiumventil besteht aus einem dünnwandigen Palladium-Röhrchen, das durch elektrisches Heizen für $H_2$ durchlässig wird, für größere Atome jedoch undurchlässig bleibt.

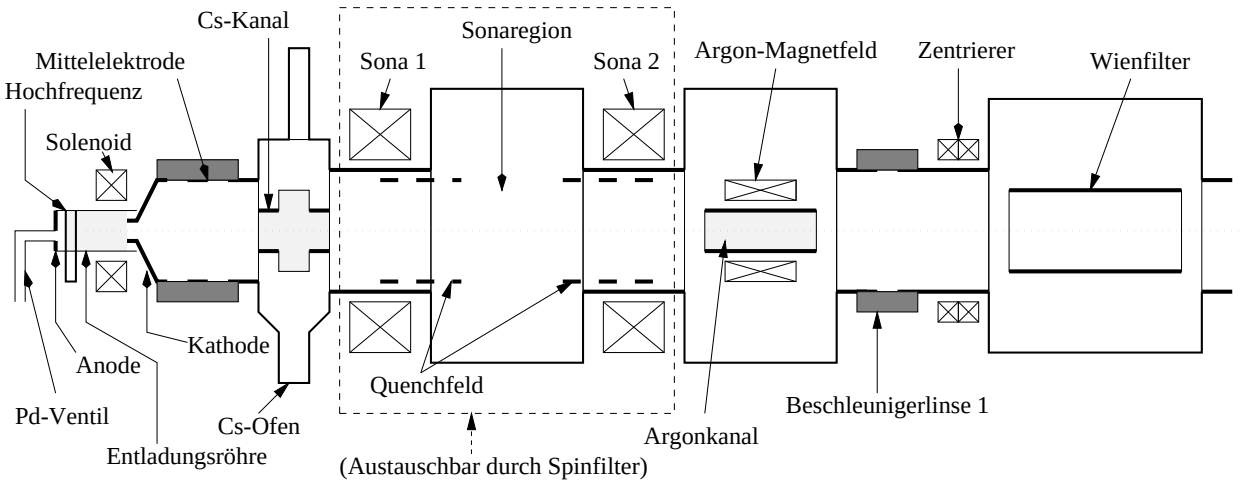


Abbildung 1.1: Schematischer Aufbau der Kölner Quelle LASCO.

Der Primärionenstrahl wird durch die Reaktion

$$H^+ + Cs^0 \rightarrow H^0 + Cs^+ + Q$$

in Cs-Dampf umgeladen. Der neutrale Strahl enthält einen gewissen Anteil an Atomen im metastabilen, d.h. für atomare Verhältnisse besonders langlebigen, 2S-Zustand. Mit Hilfe

[1] LAmbshift Source COlogne

eines E-Feldes kann durch den Stark-Effekt eine Mischung des $2S_{\frac{1}{2}}$-Zustandes und des kurzlebigen $2P_{\frac{1}{2}}$-Zustandes bewirkt werden, die die Lebensdauer des $2S_{\frac{1}{2}}$-Zustandes stark herabsetzt. Da die Lebensdauer von der Energiedifferenz der Komponenten abhängt, kann durch Anlegen eines B-Feldes eine unterschiedliche Lebensdauer der Komponenten mit $m_J = \pm\frac{1}{2}$ des Strahls erreicht werden (selektives quenching). Die entstehende Hüllenpolarisation wird dann durch Hyperfeinwechselwirkung und einen nichtadiabatischen Sona-Übergang auf den Kernspin übertragen. Alternativ zum Sona-Verfahren kann durch Austausch der entsprechenden Komponenten auf der Quelle ein Spinfilter verwendet werden (s. Abb. 1.1). Dieses erlaubt mittels verschiedener elektromagnetischer Felder die Transmission genau einer Kernspinkomponente des metastabilen Strahls; alle übrigen Zustände werden in den Grundzustand gequencht [Tr94, Fr95]. Schließlich erfolgt eine selektive Ionisation des polarisierten Metastabilenstrahls durch die Reaktion

$$H(2s) + Ar \rightarrow H^- + Ar^+,$$

um die für den Tandembeschleuniger benötigten negativen polarisierten Ionen zu erzeugen und gleichzeitig die Atome im $1S_{\frac{1}{2}}$-Zustand auszusondern.

Die Quantisierungsachse der Ionen, die bedingt durch das Argon-B-Feld innerhalb der Ionenquelle längs der Strahlachse ausgerichtet ist, kann mit Hilfe eines Wien-Filters in eine beliebige Raumrichtung gedreht werden. So läßt sich die Spinstellung am Targetort frei wählen.

| Kanal-Nr. | Sollwert | Rückmeldung |
|---|---|---|
| 1 | Anode | Anode |
| 2 | Palladium-Ventil | Sona Vakuum |
| 3 | Wienfilter B-Feld | Wienfilter B-Feld |
| 4 | Wienfilter E-Feld | Wienfilter E-Feld |
| 5 | 10 °-Ablenker | 10 °-Ablenker |
| 6 | Sona 2 | Sona 2 |
| 7 | Argon B-Feld | Argon B-Feld |
| 8 | Beschleunigerlinse 2 | Beschleunigerlinse 2 |
| 9 | Sona 1 | Sona 1 |
| 10 | Kathode | Kathode |
| 11 | Mittelelektrode | Mittelelektrode |
| 12 | Beschleunigerlinse 1 | Beschleunigerlinse 1 |
| 13 | | Strom auf Quellencup |
| 14 | Solenoid | Solenoid |
| 15 | X-Zentrierer | |
| 16 | Y-Zentrierer | |

Tabelle 1.1: Ferngesteuerte Netzgeräte und Rückmeldewerte der Quelle

Durch Umpolen, Abschalten und Variation der Sona- und Argonzellen-Magnetfelder können die Polarisationsrichtung des Strahls invertiert, die Art der Polarisation (Vektor- bzw. Tensorpolarisation für Deuteronen) geändert und die Polarisation aufgehoben werden. Es existiert ebenfalls eine Umschaltung, die den Strahl anstatt in das Beschleunigerrohr auf einen Quellen-Cup lenkt. Da die negativen Ionen der Quelle mit etwa 60 keV in den

Tandembeschleuniger eingeschossen werden müssen, befindet sich die Quelle auf einer Isolierplattform, die auf ein Potential von bis zu -80 kV gegenüber Erdpotential gebracht werden kann.

Die relevanten physikalischen Parameter der Elemente zur Strahlführung und Polarisationserzeugung werden durch die Ausgangsgrößen spannungsprogrammierbarer Netzgeräte bestimmt (siehe Tab. 1.1). Diese liefern eine ihrer Ausgangsgröße proportionale Spannung zurück. Näheres zu Aufbau und Funktion der Quelle s. [Be78, Re89].

## 1.2 Aufbau der bisherigen Steuerung

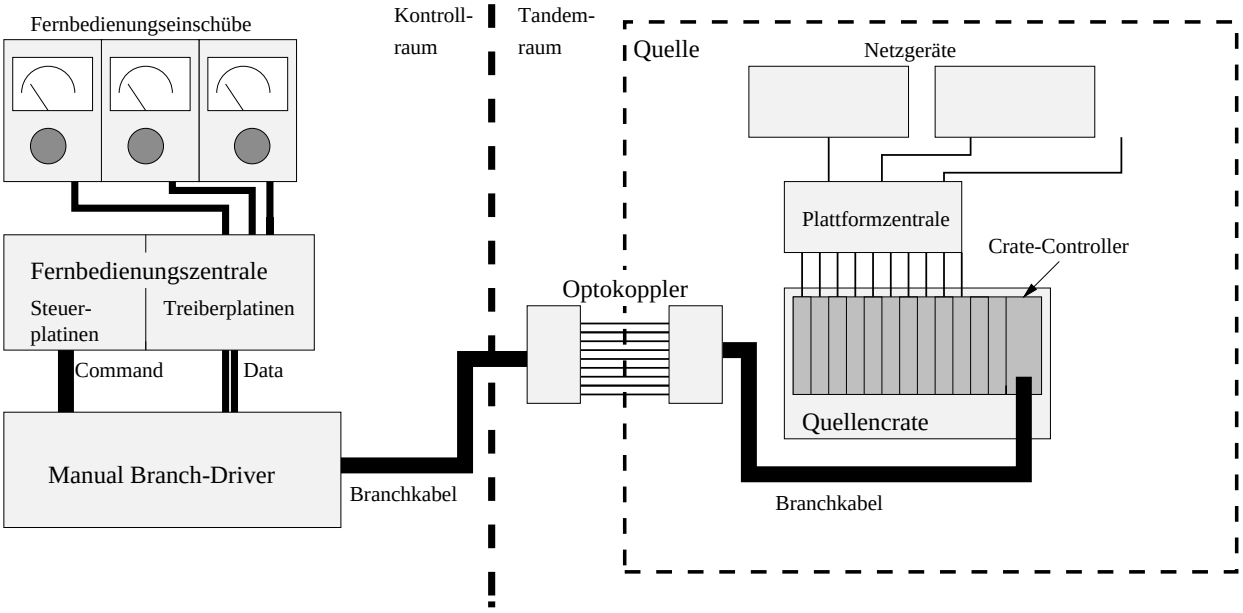


Abbildung 1.2: Das alte Fernsteuerungssystem

Um die einzelnen Quellenparameter, auch während sich die Quellenplattform unter Hochspannung befindet, bedienen zu können, wurde 1976 von R. Aniol eine Fernsteuerung für die Quelle entwickelt [An76]. Die alte wie die neue Steuerung nutzen den CAMAC[2]-Standard, ein in der Kernphysik stark verbreitetes Datenübertragungssystem zur computerunterstützten Datenaufnahme und Steuerung. (Siehe Abschnitt 4.1 )

Die alte Steuerung basiert im wesentlichen auf der digitalen Datenübertragung via CAMAC zwischen einem Kontrollpult im Tandem-Kontrollraum und den CAMAC-Bausteinen auf der Quelle, die für den Datenaustausch mit den Netzgeräten verantwortlich sind. Das Kontrollpult enthält sechzehn 10-Gang-Potentiometer und Zeigerinstrumente für die analogen und Schalter bzw. LEDs für die digitalen Quellenparameter (Polarisationsrichtung, Strahlumlenkung zu Cup oder Tandem, Quench all). Die Werte der Potentiometer werden in der Fernbedienungszentrale mit Hilfe eines ADC[3]s digitalisiert. Zum Datenaustausch zwischen den Platinen der Fernbedienungszentrale stehen zwei 16-Bit breite Busleitungen für Eingabe und Ausgabe zur Verfügung sowie ein 5-Bit-Adressbus zur Adressierung der sechzehn Analog-Kanäle. Die Daten werden über verschiedene Treiberplatinen auf den Write-Bus gelegt bzw. vom Read-Bus übernommen.

[2]Computer-Automated-Measurement-And-Control
[3]Analog Digital Converter

Der Ablauf der Datenübertragung wird durch ebenfalls in der Fernbedienungszentrale plazierte Steuerplatinen festgelegt. Diese lesen Binärbefehle aus einer Diodenmatrix, die entweder als Steuersignale für die Treiberplatinen umgesetzt oder über einen umgebauten manuellen CAMAC-Branch-Driver als CAMAC-Befehle an das Quellencrate gesendet werden.

Zur Trennung von Erd- und Quellenpotential sind die 66 parallelen Leitungspaare des CAMAC-Branch-Highways am Übergang auf die Quelle durch eine Optokopplerstrecke aus insgesamt 120 Lichtleitern unterbrochen (die bidirektionalen Leitungen benötigen 2 Lichtleiterstrecken, einige der Lichtleiterstrecken sind unbenutzt).

Bei den CAMAC-Bausteinen auf der Quelle handelt es sich um Digital-Register zur Speicherung der vom Pult übermittelten Werte sowie um einen ADC mit Multiplexer zur Digitalisierung der von den Netzgeräten zurückgemeldeten Spannungen. Die in den Registern gespeicherten Werte werden mit Hilfe einer DAC[4]-Platine in Steuerspannungen für die Netzgeräte umgewandelt.

Die Netzgeräte können entweder intern über ein Potentiometer auf der Frontplatte oder extern über eine Steuerspannung geregelt werden. Die DAC-Platinen einiger Netzgeräte enthalten zusätzlich einen Wertübernahmemechanismus, der eine Umschaltung zwischen den beiden Betriebsarten erst dann vornimmt, wenn extern und intern eingestellter Wert in etwa gleich sind.

## 1.3 Modernisierung der Steuerung

Die Modernisierung der Steuerung durch ein rechnergestütztes System bietet eine Reihe von zusätzlichen Vorteilen und Erweiterungsmöglichkeiten. Darüber hinaus haben sich verschiedene Teile der alten Steuerung als störanfällig erwiesen.

### Vorteile und zusätzliche Möglichkeiten einer rechnergestützten Steuerung

- Überwachung und Bedienung der Quelle über das Rechner-Netzwerk
- Möglichkeit der Protokollierung und Reproduktion der Quellenparameter
- Optimierung des Steuerungsablaufes durch ein komplexes Steuerprogramm
- Mögliche Automatisierung auch komplexer Steuer- und Überwachungsaufgaben
- Vereinfachung der Fehlersuche durch umfangreiche Protokoll- und Visualisierungsmöglichkeiten
- Höhere Betriebssicherheit durch automatisierte Kontrollmechanismen
- Komfortablere Einstellmöglichkeiten durch Vorbearbeitung der Sollwerte (z.B. Eichung, Untersetzung, Sollwertbegrenzung)

[4]Digital Analog Converter

### Nachteile der alten Steuerung

- Die Steuerungsplatinen der Fernbedienungszentrale haben sich als störanfällig herausgestellt. Dies ist größtenteils auf die durch das Alter der Platinen hervorgerufenen Kontaktprobleme und den Verschleiß einzelner Bauelemente zurückzuführen. Mit den heutigen technischen Möglichkeiten läßt sich prinzipiell eine erhöhte Zuverlässigkeit erzielen.

- Da das alte System parallele CAMAC-Leitungen benutzt, mußten für sehr viele einzelne Leitungen Optokopplerstrecken eingerichtet werden, was die Fehleranfälligkeit erhöht und die Wartung des Systems aufwendig macht.

- Die alte Steuerung ist nur schwer zu erweitern und neuen Aufgaben anzupassen.

- Elektrische Überschläge im Tandem können zum Ausfall des alten Crate-Controllers führen, so daß die komplette Fernsteuerung nicht mehr nutzbar ist. Da der alte Crate-Contoller nur von Hand wieder eingeschaltet werden kann, muß die Quelle betreten und daher das Potential der Quelle auf Erdpotential gebracht werden. Dies kann jedoch für einige Netzgeräte nur dann gefahrlos geschehen, wenn die Beschleunigerlinse 2 zurückgefahren wurde, was aber ohne Fernsteuerung nicht möglich ist.

- Bei der alten Steuerung müssen die auf den Netzgeräten der Quelle intern eingestellten Werte beim Übergang zum ferngesteuerten Betrieb per Hand nachgeführt werden. Einige der Netzgeräte besitzen zu diesem Zweck eine Nachführungseinrichtung, die die Netzgeräte nur dann auf externen Betriebsmodus schalten läßt, falls der intern eingestellte Wert mit dem durch die Fernsteuerung angeforderten Wert in etwa übereinstimmte. Dieses Verfahren ist zum einen unbequem, zum anderen recht ungenau, da der Strahl auf einzelne Parameter schon innerhalb der Toleranzgrenzen der Nachführeinrichtung sehr empfindlich reagiert.

- Im alten System wird pro Datenübertragungszyklus nur jeweils einer der 16 Kanäle des Quellen-ADCs ausgelesen. Dies führt zu einer schlechten zeitlichen Auflösung in der Anzeige der Zeigerinstrumente des Kontrollpultes.

Für die Strahljustierung hat sich die Bedienung über Potentiometer als nützlich erwiesen. Daher sollten die Bedienelemente des alten Kontrollpultes beibehalten werden. Auch die CAMAC-Bausteine der Quelle sollten bei der neuen Steuerung weiterhin verwendet werden, da diese zuverlässig arbeiten und eine Erneuerung in der ersten Ausbaustufe nicht notwendig erscheint.

### Anforderungen an die neue Steuerung

Für ein modernisiertes Fernsteuerungssystem sind demnach folgende Anforderungen zu erfüllen:

- Der Steuerungsrechner ist an das Instituts-Rechnernetz bzw. Internet anzubinden, um eine Steuerung und Überwachung der Quelle von beliebigen Terminals zu ermöglichen. Dies ermöglicht u.a. den für Wartungsarbeiten sehr wichtigen Betrieb der Fernsteuerung direkt von der Quellenplattform aus.

- Die Bedienelemente des alten Kontrollpultes sollen erhalten bleiben, da sie sich als zweckmäßig herausgestellt haben. Für den Benutzer ergibt sich auf diese Weise kaum eine Umstellung. Die störanfälligen Steuerplatinen sollen jedoch nicht mehr verwendet werden.

- Die Steuerung soll jederzeit in ihren alten Zustand rückbaubar sein, um die Einsatzfähigkeit der Quelle in jeder Phase der Entwicklung zu gewährleisten. Nach Fertigstellung des neuen Systems kann die alte Übertragungsstrecke entfernt werden.

- Die parallele Optokopplerstrecke soll durch ein serielles System ersetzt werden. Dadurch reduziert sich die Zahl der benötigten Lichtleiter auf 2.

- Die Software soll Möglichkeiten zur Steuerung durch externe Programme bieten. Dies dient zur teilweisen Automatisierung der Steuerung der Quelle während der Strahlzeiten durch das Analysator-Meßprogramm.

- Die Software muß einerseits auf Aktionen des Benutzers sofort reagieren, darf andererseits jedoch nicht zuviel Rechenzeit in Anspruch nehmen, um die Ausführung anderer Programme nicht zu beeinträchtigen.

- Möglichkeiten zur Abspeicherung und Protokollierung der Quellendaten müssen gegeben sein.

- Die Übernahme der auf den Quellennetzteilen intern eingestellten Werte durch die Fernsteuerung soll automatisiert werden.

- Die Einbindung einer grafischen Benutzeroberfläche für das Steuerprogramm soll ermöglicht werden.

- Überwachungsmechanismen zur Erkennung und Behandlung von Übertragungsfehlern und Störungen auf der Quelle sind vorzusehen.

# Kapitel 2

# Ausführung der neuen Steuerung

Im folgenden soll ein kurzer Überblick über Aufbau und Möglichkeiten der neuen Steuerung gegeben werden. Die technischen Details werden in den folgenden Kapiteln behandelt. Aus Gründen der Übersichtlichkeit werden die wesentlichen Punkte des Überblicks dort nochmals aufgeführt. Eine ausführliche Betriebsanleitung ist im Anhang enthalten.

## 2.1 Hardware

**Rechner:** Zentrales Element der Steuerung ist ein Personal Computer unter dem UNIX-kompatiblen Betriebssystem LINUX, der über eine Anbindung an das Rechnernetz des Institutes verfügt. Der PC besitzt eine spezielle Interface-Karte, mit der CAMAC-Befehle an ein Steuercrate übermittelt werden können.

**Serieller Highway:** Die Übertragung der Daten zur Quelle geschieht über einen Bit-seriellen CAMAC-Highway. Das Senden der Befehle erfolgt durch einen im Steuercrate befindlichen seriellen Highway-Driver. Auf der Quelle werden die Befehle von einem seriellen Crate-Controller empfangen und an die CAMAC-Module im Quellencrate weitergegeben, die die Netzgeräte der Quelle steuern. Der serielle Highway ist als Ring konzipiert, so daß jedes Datenpaket zum Absender zurückgelangt und dort auf Übertragungsfehler geprüft wird.

**U-Port-Adapter:** Zur Trennung von Erd- und Quellenpotential ist der serielle Highway elektrisch durch eine serielle Lichtleiterstrecke unterbrochen. Die optische Übertragung auf dieser Strecke wird von zwei U-Port-Adaptern geleistet, die die Wandlung und Kodierung des Datensignals vornehmen. Zur Erhöhung der Übertragungssicherheit existiert auf der Lichtleiterstrecke ein zusätzlicher Backup-Loop, der im Falle einer Störung des Main-Loops die Übertragung des Datensignals übernimmt.

**Kontrollpult:** Die Anbindung des Kontrollpultes der alten Steuerung geschieht durch ein $2 \times 24$-Bit CAMAC-IO-Register. Die Steuerplatinen und der Branch-Driver der alten Steuerung wurden entfernt, ihre Funktion wird nun durch den Rechner übernommen. Beibehalten wurden lediglich die Treiberplatinen.

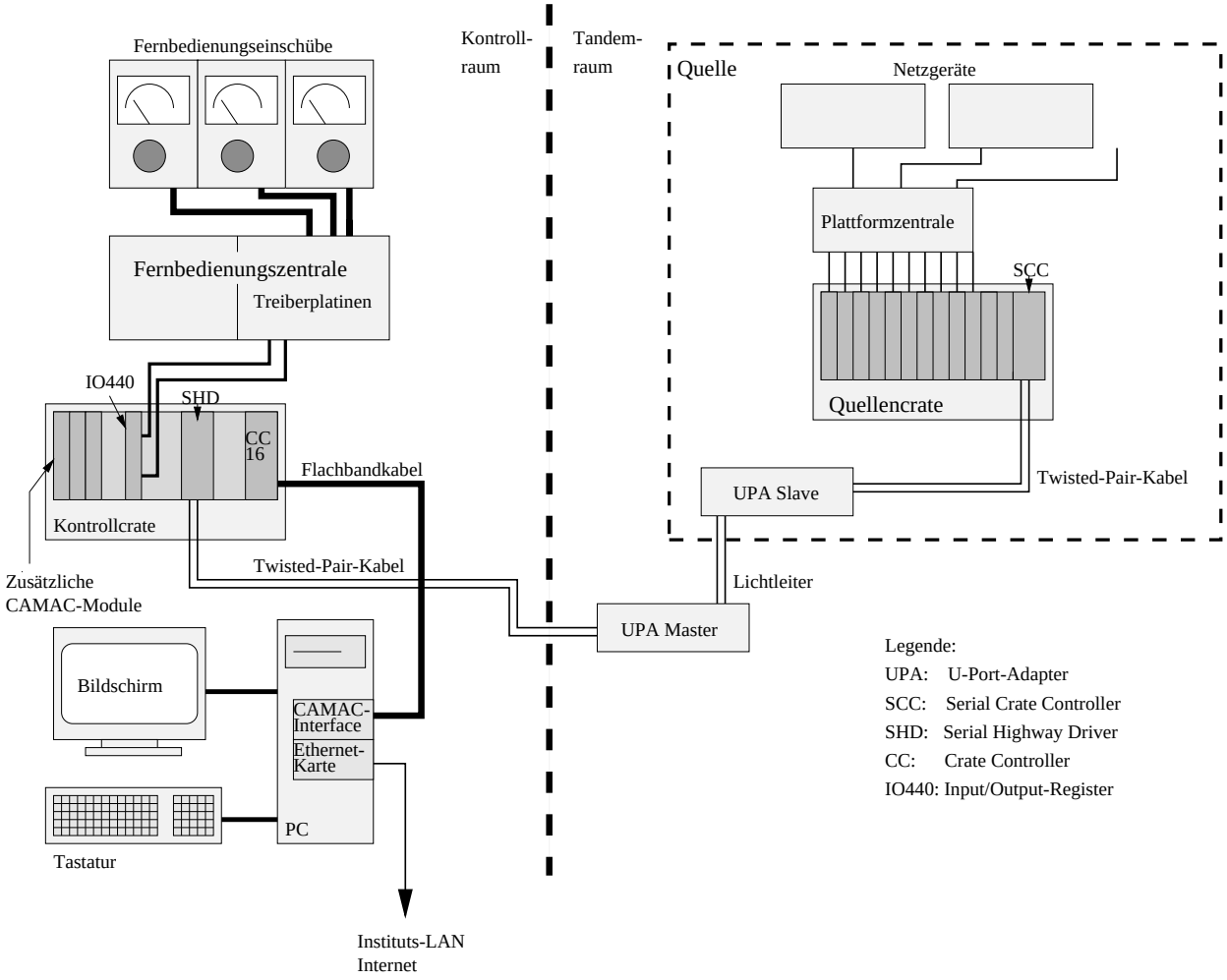


Abbildung 2.1: Die neue Steuerung

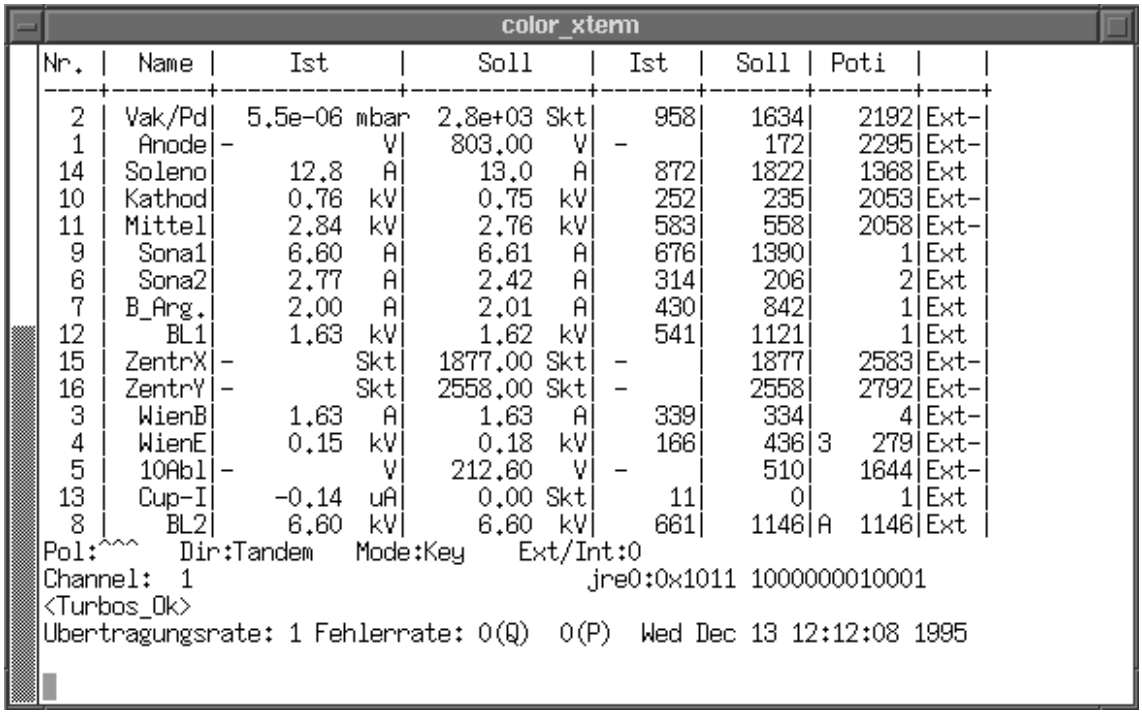


Abbildung 2.2: Bildschirmausgabe des Steuerprogrammes *piq*

## 2.2 Software

**Programme:** Die Steuersoftware besteht aus dem zentralen Steuerprogramm *piq* (Abb. 2.2), das nahezu alle Steueraufgaben erledigt und eine Bedienung der Quelle über ein Textterminal oder das Kontrollpult ermöglicht. Desweiteren existieren Client-Programme, die direkt mit dem Steuerprogramm kommunizieren und Aufgaben wie der Steuerung über eine grafische X-Windows-Oberfläche (das Programm *xpiq*, Abb. 2.3) oder über eine Remote-Shell dienen. Ein weiteres Hilfsprogramm dient der Online-Überwachung.

**Betriebsmodi:** Durch Umschalten zwischen drei Betriebsmodi läßt sich das Steuerprogramm über das Kontrollpult oder ein Terminal bedienen und alternativ die Steuerung zur Verhinderung von Fehlbedienungen sperren.

**Bedienung über das Kontrollpult:** Die Quelle kann wie bisher über das Kontrollpult gesteuert werden. Die Sollwerte der Netzgeräte werden über Potentiometer eingestellt, die Istwerte werden auf Zeigerinstrumenten angezeigt.

Wesentlicher Unterschied zur alten Steuerung ist, daß vom Steuerprogramm nicht die absolute Stellung der Potentiometer , sondern Änderungen der Potentiometer - Positionen verwertet werden. Dadurch ist eine problemlose Umschaltung zwischen den verschiedenen Betriebsmodi und eine automatische Übernahme der auf der Quelle intern eingestellten Werte möglich. Mit geringem Aufwand wird auf diese Art die Funktionsweise von teuren Digital-Potentiometern simuliert.

**Bedienung über ein Textterminal:** Die Bedienung des Steuerprogrammes kann über ein einfaches Textterminal erfolgen. Sämtliche relevanten Quellenparameter werden

auf dem Bildschirm angezeigt und kontinuierlich aktualisiert. Die Bedienung erfolgt direkt über die Tastatur, wobei die elementaren Funktionen des Programmes über Funktionstasten abgerufen werden können. Die erweiterten Funktionen können in einer Kommandozeile eingegeben werden.

**Bedienung über ein X-Terminal:** Durch ein zusätzliches Client-Programm läßt sich das Steuerprogramm über eine grafische Benutzeroberfläche (Abb. 2.3) bedienen. Alle Funktionen sind per Mausklick abrufbar.

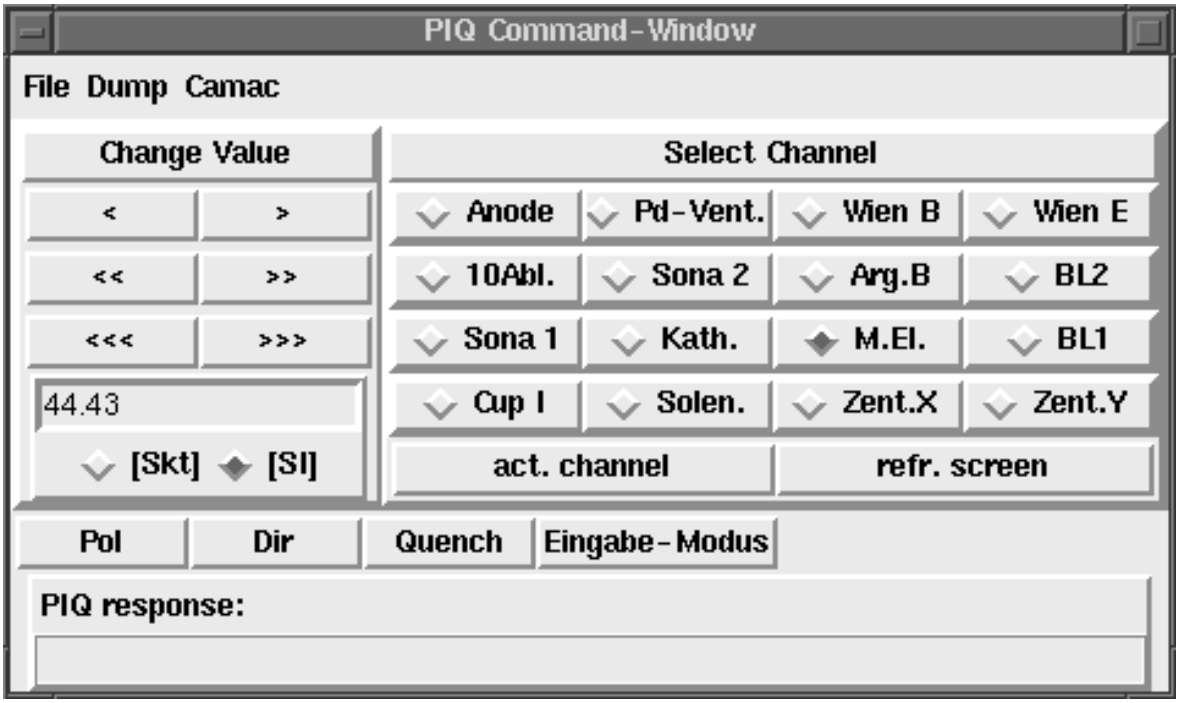


Abbildung 2.3: Die grafische Benutzeroberfläche *xpiq*

**Werteübernahme von der Quelle:** Die Übernahme der auf den Quellennetzgeräten intern eingestellten Werte ist nun weitestgehend automatisiert.

**Client-Programme:** Das Steuerprogramm *piq* läßt die externe Steuerung über Client-Programme zu. Alle Funktionen der Steuerung lassen sich über ein Software-Interface von externen Programmen als Befehl zum Programm *piq* senden. U.a. wurde ein Interpreter für die grafische Programmiersprache Tcl/Tk entwickelt, der einen erweiterten Befehlssatz zur Steuerung der Quelle zur Verfügung stellt. Mit Hilfe dieses Interpreters wurde die grafische Benutzeroberfläche des Steuerprogrammes erstellt.

**CAMAC-Kommandos:** Über das Steuerprogramm können parallel zur Übertragung der Steuerungsbefehle beliebige CAMAC-Befehle an das Steuercrate oder über den Serial Highway übertragen werden. Die Befehle können aus externen Programmen, über Tcl/Tk, durch Eingabe einer Kommandozeile im Steuerprogramm oder ein Untermenü in der X-Kommandooberfläche (Abb. 2.4) eingegeben werden.

**Datenaufnahme:** Das Steuerprogramm bietet die Möglichkeit der automatischen Protokollierung der Quellendaten zur späteren Auswertung. Ebenso lassen sich Einstellungen der Quelle zur späteren Reproduktion abspeichern.

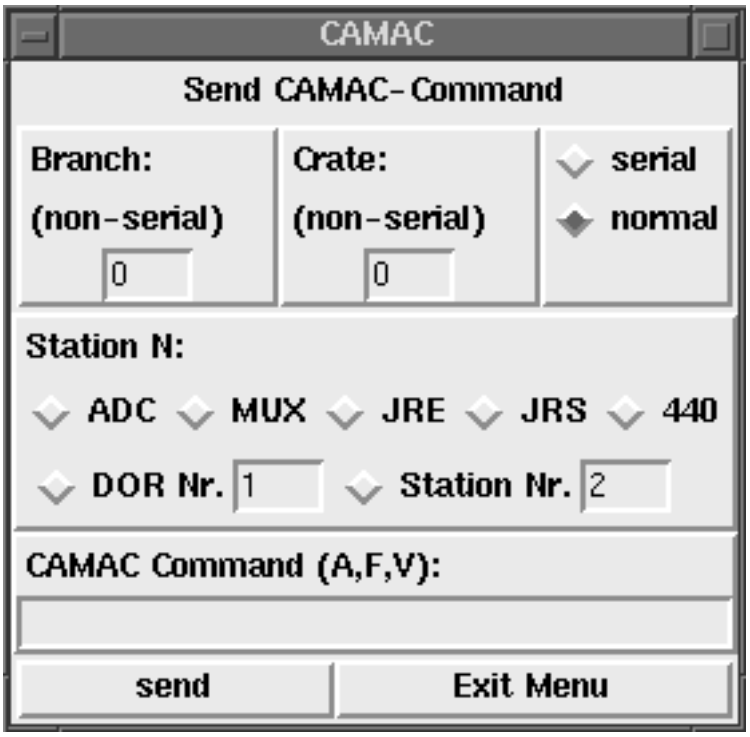


Abbildung 2.4: Das Eingabefenster für CAMAC-Befehle des Programmes *xpiq*. Es dient hauptsächlich der Fehlersuche und Testzwecken.

**Konfigurationsmöglichkeiten:** Nahezu alle systemspezifischen Parameter des Programmes lassen sich durch ein Setup-Skript frei konfigurieren. Die freien Bits des digitalen Quelleneingaberegisters lassen sich ebenfalls durch Festlegung im Konfigurationsskript für Meldungen der Quelle nutzen. Durch verschiedene Konfigurationsskripts ist eine problemlose Umstellung der Steuerung zwischen verschiedenen Quellenkonfigurationen wie z.B. Spinfilter- und Sonabetrieb möglich.

Der Tcl/Tk-Interpreter bietet dank der Effektivität undleichten Erlernbarkeit der Programmiersprache Tcl/Tk eine Möglichkeit der einfachen Automatisierung auch komplexer Abläufe, ohne daß eine Einarbeitung in die innere Struktur des Steuerprogrammes nötig ist.

Überschreitet der Istwert eines Netzgerätes eine vom Benutzer festgelegte Schwelle, wird ein ebenfalls frei konfigurierbares UNIX Shellskript ausgeführt, in dem sich Steueranweisungen für die Quelle und/oder Meldungen an festgelegte Benutzer (z.B. über E-Mail) festlegen lassen.

# Teil II

# Hardware

# Kapitel 3

# Überblick über die Hardware

Da im neuen Steuerungssystem viele Aufgaben von der Software erledigt werden, für die in der alten Steuerung Hardwarelösungen existierten, bestand der Schwerpunkt der Elektronikarbeiten nicht in der Neuentwicklung von Hardwarekomponenten, sondern in der Integration der vorhandenen Geräte in das Gesamtsystem und der Schaffung von Möglichkeiten zur softwaremäßigen Steuerung der Geräte. Trotz des relativ geringen handwerklichen Aufwandes war eine tiefgehende Einarbeitung in Aufbau und Funktionsweise der alten Steuerelektronik notwendig, um die vorhandenen Komponenten für die neue Steuerung nutzen zu können. Auch wurden zahlreiche Reparaturarbeiten an den Platinen durchgeführt, die hier nicht im einzelnen aufgeführt werden sollen.

Entsprechend den Anforderungen an die neue Fernsteuerung mußten folgende Hardware-Probleme gelöst werden:

## 3.1 Problemstellungen

1. Auswahl eines geeigneten Rechnersystems und Anbindung an CAMAC. Anbindung an das Institutsnetz (LAN[1]).

2. Ersetzen des parallelen Branchkabels und der parallelen Optokopplerstrecke durch ein serielles System.

3. Einbindung der Bedienelemente des alten Kontrollpultes, jedoch Ersetzung der Steuerplatinen.

4. Option des einfachen Rückbaus der Steuerung in den ursprünglichen Zustand.

## 3.2 Problemlösungen

1. Die Steuerung erfolgt durch einen PC unter dem Betriebssystem LINUX, der über ein CAMAC-Interface verfügt. Das Interface spricht über einen eigenen Crate-Controller ein Steuercrate an, welches CAMAC-Bausteine zur Ansteuerung des seriellen Highways und des Kontrollpultes enthält.

2. Die Datenübermittlung zur Quelle geschieht über einen seriellen Highway statt wie bisher über einen parallelen Branch-Highway. Der Optokoppler wurde ebenfalls durch

[1] Local Area Network

ein serielles System ersetzt. Das löst nicht nur das Problem der Optokopplerstrecke, sondern macht auch aufgrund der erheblich geringeren Kabelkosten eine Verkabelung über größere Distanzen möglich. Der Highway wird über einen Serial Highway-Driver, der sich ebenfalls im Steuercrate befindet, angesprochen.

3. Im Kontrollpult wurden die fehleranfälligen Steuerplatinen entfernt und nur die Treiberplatinen beibehalten. Diese werden nun vollständig über ein im Steuercrate befindliches IO-Register durch den Rechner gesteuert.

4. Durch einfaches Umstecken zweier Steckverbindungen, Wiedereinsetzen der Steuerplatinen und Austausch des Crate-Controllers auf der Quelle kann die Steuerung in ihren alten Zustand zurückgebaut werden.

# Kapitel 4

# Aufbau des Serial Highway

## 4.1 Das CAMAC-System

Der CAMAC-Standard [Ie71, Le79] ist ein in den sechziger Jahren speziell für kernphysikalische Meß- und Steuerungsaufgaben eingeführter Schnittstellenstandard, der Spezifikationen über den zwischen den Geräten verlaufenden Datenbus und über die physikalischen Abmessungen der Bauteile enthält.

Die verschiedenen Module des CAMAC-Systems, wie Ein/Ausgaberegister oder ADCs, werden in Crates angeordnet, die mechanische Einschubplätze für die Module enthalten und auch die nötigen Versorgungsspannungen zur Verfügung stellen. Ein CAMAC-Crate bietet 24 Modulen einfacher Breite Platz und stellt ihnen stabilisierte Spannungen ($\pm 24V, \pm 6V$, meist auch $\pm 12V$) zur Verfügung. Die Module werden durch den Crate-internen Bus, den CAMAC-Dataway, angesprochen und über die Nummer ihres Einschubplatzes innerhalb des Crates adressiert.

Der CAMAC-Dataway-Bus enthält 3 Steuerleitungen (initialisieren, sperren, clear) , 5 Befehlsleitungen, 4 Adressleitungen, je 24 Lese- und Schreibleitungen, 2 Timingsignale und 4 Statusleitungen (Tab. 4.1). Die Signale Q und X dienen als Antwortsignale für die einzelnen CAMAC-Funktionen, das LAM (Look-At-Me)-Signal dient dazu, das CAMAC-System auf ein bestimmtes Modul aufmerksam zu machen, etwa wenn neue Daten anliegen.

Jedes Crate wird über einen Crate-Controller gesteuert, der als Modul doppelter Breite in den Positionen 25 und 26 des Crates eingeschoben und über eine einstellbare Crate-Adresse adressiert wird. Der Crate-Controller aktiviert nach Dekodierung der Moduladresse aus dem CAMAC-Highway das entsprechende Modul über eine der Signalleitungen N, die vom Crate-Controller zu jeweils einem der Module führen.

Der CAMAC-Highway ist der Datenbus, über den die verschiedenen Crates mit der Steuerung kommunizieren. Er kann verschiedene Ausführungen haben. Das in der alten Steuerung verwendete System benutzt den Branch Highway, in dem alle Dataway-Signale sowie Branch-, Crate- und Moduladresse parallel geführt werden. Ein neueres System, das in der neuen Steuerung verwendet wird, ist der Bit-serielle Highway, in dem die Daten über einen seriellen Loop übermittelt werden.

Schnittstelle zwischen dem steuernden Computer und dem entsprechenden Highway ist ein Highway-Driver, der entweder vom Rechner direkt angesteuert wird, oder, wie im Falle des seriellen Highway-Drivers der neuen Steuerung, selbst ein CAMAC-Modul ist und als solches über CAMAC angesprochen wird.

Ein CAMAC-Befehl besteht zunächst aus Branch-, Crate- und Moduladresse (B, C, N) des angesprochenen Moduls. Hinzu kommen eine 4-Bit-Subadresse A, die nicht standardi-

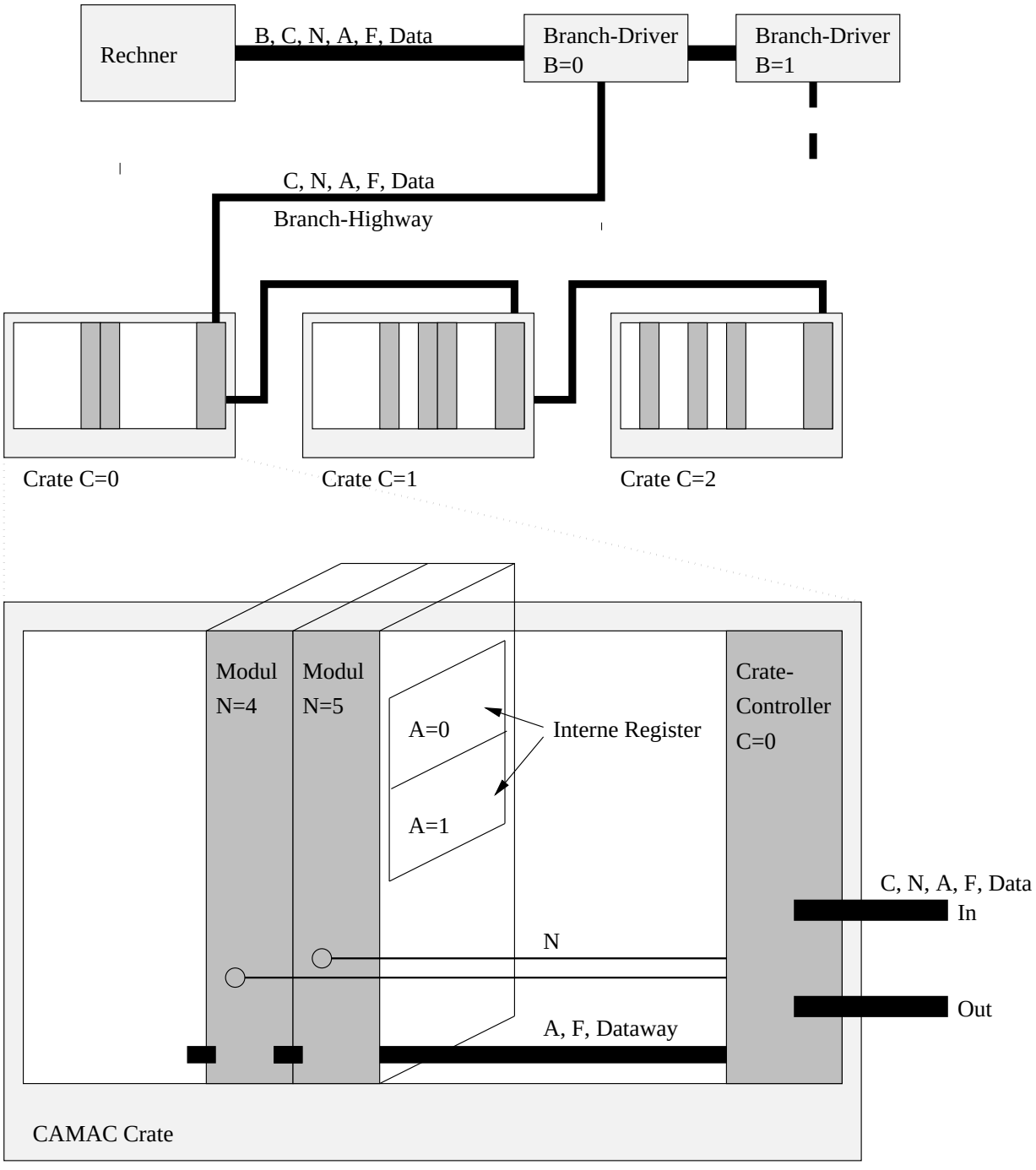


Abbildung 4.1: Aufbau und Adressierung eines CAMAC-Systems (schematisch)

| Art | Name | Bezeichnung | Funktion |
|---|---|---|---|
| Steuer- | Initialize | Z | Setzt alle Module in definierten Zustand |
| signale | Inhibit | I | Sperrt bestimmte Funktionen |
| | Clear | C | Löscht alle Register |
| Adresse | Station Nr. | N | Selektiert Modul im Crate |
| | Subadress | A1,2,4,8 | Interne Adresse des Moduls |
| Funktion | Function Code | F1,2,4,8,16 | Bestimmt die auszuführende Funktion |
| Daten | Read Bus | R1-24 | Datenübermittlung zu Crate-Controller |
| | Write bus | W1-24 | Datenübermittlung zu Modul |
| Timing | Strobe | S1,S2 | Werden zum Timing der Module benutzt |
| Status- | Look at me | LAM, L | Modul fordert Bedienung |
| Signale | Q Response | Q | Antwort auf bestimmte Anfragen |
| | Accepted | X | Modul führt gegebenes Kommando aus |
| | Busy | B | Dataway-Operation wird ausgeführt |

Tabelle 4.1: CAMAC Dataway-Signale

siert ist und innerhalb des Moduls verwendet wird, sowie die Funktionsnummer F, die die Werte 0 bis 31 annehmen kann. Die Funktionen sind dahingehend standardisiert, daß Bit 4 angibt, ob es sich bei dem Befehl um ein reines Kommando handelt, bei denen die Datenleitungen nicht benutzt werden, oder um eine Read- bzw. Write Funktion. Im letzteren Falle bedeutet Bit 5 = 1 ein Write, Bit 5 = 0 ein Read. Somit stellen Fkt. 0 bis 7 Read-Funktionen, Fkt. 16 bis 23 Write-Funktionen und alle übrigen reine Kommandofunktionen dar.

## 4.2 Rechner

Als Rechnerplattform wurde ein PC mit 486 DX4-100-Prozessor gewählt. Der Rechner verfügt über 16 MB Arbeitsspeicher und 560 MB Plattenkapazität. Diese Ausstattung gewährleistet, daß das Betriebssystem LINUX effizient eingesetzt werden kann. Auch bei hoher Belastung lassen sich mehrere Prozesse nebeneinander nutzen, ohne daß sich dies für den Benutzer negativ in der Reaktionszeit auswirken würde.

Die Anbindung des PCs an CAMAC geschieht durch eine Kombination aus AT-Bus-Steckkarte und Crate-Controller der Firma Wiener [Wi01], die mit einem 16-adrigen Flachbandkabel verbunden sind. Die Kabellänge kann laut Hersteller bis zu 30 m betragen, im vorliegenden System beträgt sie 10 m. Auf die Ansteuerung und Programmierung des Interface wird in Abschnitt 7.2.1 eingegangen.

Zur Anbindung an das LAN verfügt der Rechner über eine Ethernetkarte.

## 4.3 Der Serial Highway

Der Serial Highway besteht aus dem Serial Highway Driver (SHD), zwei U-Port-Adaptern (UPA) und einem seriellen Crate-Controller (SCC) der Firma Hytec[Hy01, Hy02, Hy03].

Die Schnittstelle zwischen SHD und SCC ist ein im RS422-Standard arbeitender "defined port" oder D-Port. Dieses differentielle Verfahren erlaubt jedoch nur eine geringe Potentialdifferenz zwischen den Geräten. Zur Überwindung der 60 kV Potentialdifferenz zwischen

Quelle und Erdpotential werden die Signale des D-Ports deshalb durch die U-Port-Adapter auf einen "undefined port" oder U-Port auf Lichtleiterbasis umgesetzt.

Die CAMAC-Befehle zur Steuerung des Quellen-Crates werden bitweise vom SHD an den Master-UPA geschickt und von dort über eine Lichtleiterstrecke an den Slave-UPA auf der Quelle. Dieser steuert den SCC im Quellencrate, der die Befehle an die einzelnen Bausteine im Crate weitergibt. Zur Übertragung des Ergebnisses des CAMAC-Befehls und zur Fehlerkorrektur wird der Signalweg anschließend zum Ausgangspunkt zurückgeführt.

### 4.3.1 Technischer Aufbau und Konfiguration

#### Signalweg

SHD und SCC können sowohl in einem Bit-seriellen als auch in einem Byte-seriellen Modus arbeiten. Die U-Port-Adapter erfordern jedoch einen Betrieb des Serial Highway im Bit-seriellen Modus.

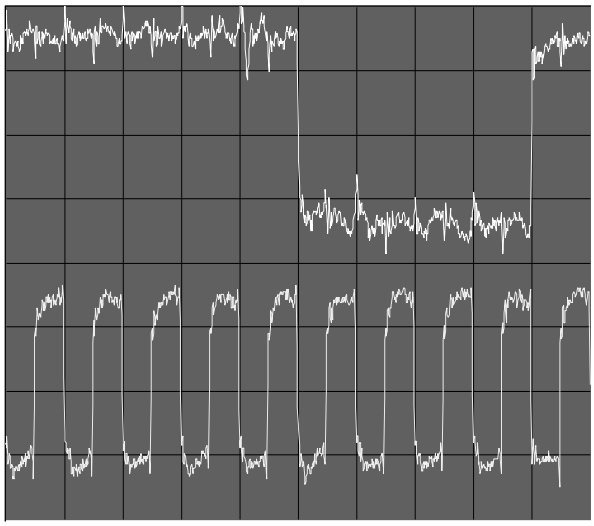

Abbildung 4.2: Data- und Clock-Signal des Serial Highway bei 1 MHz.
X-Achse: 1 Einheit = $1\mu sec$. Y-Achse: 1 Einheit = $1V$.

Übertragen werden ein Data-Signal und ein Clock-Signal, das zur Identifizierung der einzelnen Bits des Data-Signals notwendig ist (Abb. 4.2). Die Übertragung der Signale erfolgt differentiell über Twisted Pair-Kabel. Für die Definition des logischen Zustandes des Signals ist lediglich die Spannungsdifferenz zwischen beiden Adern des verdrillten Kabelpaares ausschlaggebend; eine induzierte elektromagnetische Störung wird sich in dieser Differenz wegheben und hat keinen Einfluß auf den logischen Zustand. Auf diese Weise läßt sich eine sehr große Übertragungsstrecke störungsfrei überbrücken.

Der Highway ist als Ring aufgebaut, d.h. das Signal wird vom letzten Gerät wieder an den SHD zurückgeführt. Für diese Rückführung ist normalerweise ein zweiter Kabelstrang vorgesehen, im Bit-seriellen Modus läßt sich die Verkabelung jedoch vereinfachen: Die D-Ports der Geräte enthalten 2 unbelegte Bus-Leitungspaare, die je 2 Pin-Paare des In- mit denen des Out-Ports verbinden. Mittels eines Turn-Round-Connectors (TRC) können die

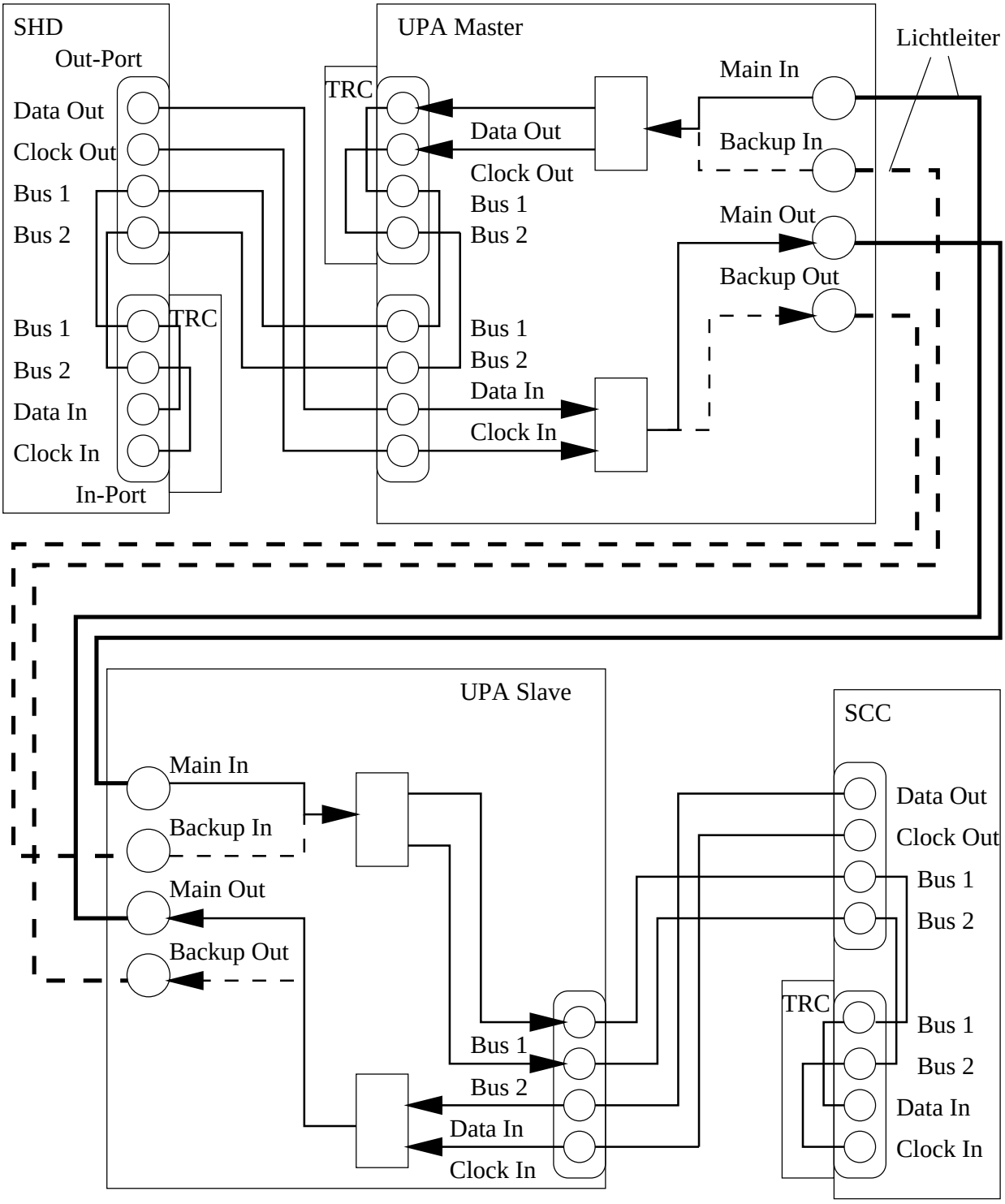


Abbildung 4.3: Skizze der seriellen Übertragungsstrecke des Serial Highways mit Turn-Round-Connectors.

beiden Signale Bit und Clock über die beiden Busleitungen in nur einem 4-paarigen statt in zwei 2-paarigen Kabeln hin- und zurückgeführt werden. Dabei erhalten der Input von Gerät 1 und der Output von Gerät 2 einen Turn-Round-Connector (TRC), der das Bus 1- mit dem Bit- und das Bus 2- mit dem Clock-Signal verbindet, so daß der Output von Gerät 2 wieder am Input von Gerät 1 anliegt und so der Loop geschlossen wird (Abb. 4.3). Diese Methode ist auf der Übertragungsstrecke zwischen SHD und UPA1 optional, jedoch zwischen UPA2 und SCC zwingend, da der UPA2 für diese Verkabelungsart ausgelegt ist und nur einen In-Port besitzt.

Die U-Port-Adapter bilden aus dem Data- und dem Clock-Signal ein einziges Signal, indem sie jedes Bit nach der Hälfte des Zeittaktes invertieren (Manchester Self-Clocking Return-to-Zero Code [Hy03]). Anhand der zeitlichen Abstände bis zur Inversion des Signals können vom Empfänger Takt- und Dateninformation wieder extrahiert werden. Durch dieses Verfahren wird für jede Signalstrecke lediglich ein Lichtleiter benötigt. Zur Verhinderung des Zusammenbruchs des Systems bei Ausfall eines Lichtleiters ist ein zusätzlicher Backup-Loop aus 2 Lichtleitern vorhanden.

### U-Port-Adapter

Die U-Port-Adapter können über verschiedene Parameter gesteuert werden, die den Weg des Signals über Main- und Backup-Leitungen bestimmen (Abb. 4.4). Der gewählte Signalweg wird duch LEDs auf der Frontseite der UPAs angezeigt. Mit Hilfe des oberen Switch-Panels auf der Frontplatte der UPAs kann eingestellt werden, welche Parameter manuell (mit Hilfe des unteren Switch-Panels) und welche automatisch gesteuert werden sollen. Die Nummern der jeweiligen Switches entsprechen den Nummern auf der LED-Schalttafel, so daß die Bedienung leicht ersichtlich ist.

Von besonderer Bedeutung sind die Sync-LEDs, die anzeigen, daß ein logisch und elektrisch korrektes Signal am Lichtleitereingang des U-Port-Adapters anliegt. Im fehlerfreien Betrieb müssen bei beiden UPAs die Sync-LEDs des Main-Loops leuchten, ansonsten liegt eine Störung vor. Der Signalweg muß im übrigen so verlaufen, daß ein geschlossener Loop entsteht, der Higway-Driver und Crate einschließt.

| Bit 2<br>Dip 6 | Bit 1<br>Dip 5 | Bit 0<br>Dip 4 | Verwendeter<br>In-Loop | Verwendeter<br>Out-Loop |
|---|---|---|---|---|
| On | On | On | Main | Main (Default-Konfiguration) |
| On | On | Off | Main | Main & Backup (disabled) |
| On | Off | On | Main | Main & Backup |
| On | Off | Off | Main | Backup & Main (disabled) |
| Off | On | On | Backup | Main & Backup |
| Off | On | Off | Backup | Main & Backup (disabled) |
| Off | Off | On | Backup | Backup |
| Off | Off | Off | Backup | Backup & Main (disabled) |

Tabelle 4.2: Signalweg-Konfigurationen des Master UPA. Bit 0,1,2 liegen auf den Dip-Schaltern Nr.4,5,6 des Kodiersteckers. On=0, Off=1.

Der Master-UPA läßt nur zwei manuelle Einstellungen zu. Die Loopkonfiguration wird über einen Auxiliary-Port gesteuert. Die acht möglichen Konfigurationen werden als 3-Bit-Wert an drei Pins des AUX-Ports gegeben und können dem Handbuch oder Tabelle 4.2

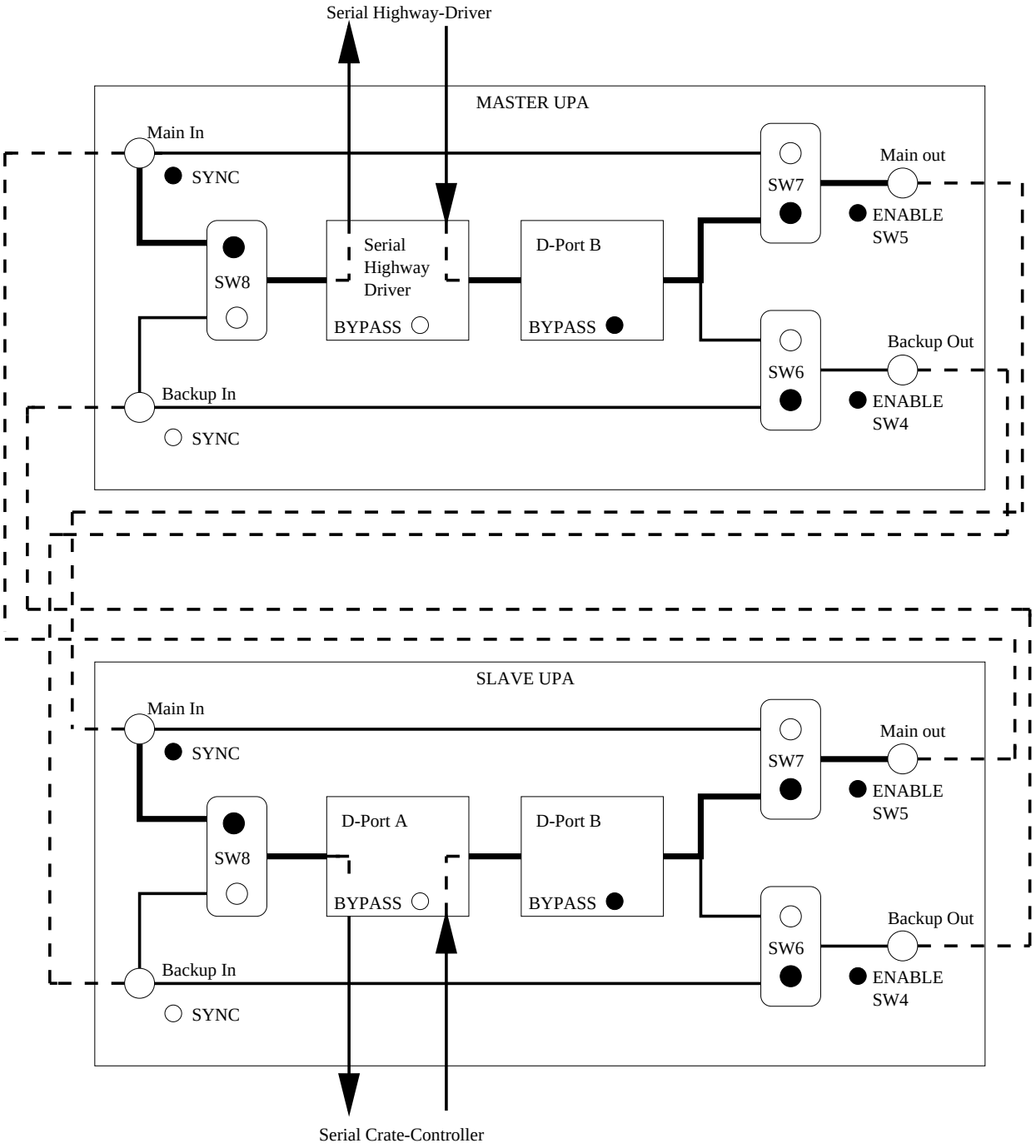


Abbildung 4.4: Signalweg und Konfiguration der U-Port-Adapter. Das vom Highway-Driver am UPA-Master ankommende Signal wird durch die Multiplexer SW6 und SW7 auf die beiden Lichtleiterschleifen (Main- und Backup-Loop) gelegt. Über die Lichtleiterstrecke gelangen die Signale in die Eingänge des UPA-Slaves. Die SYNC-LEDs am Eingang zeigen an, ob das entsprechende Signal empfangen wird. SW8 legt nun eines der beiden Signale auf die Ausgangs-D-Ports. Von D-Port A gelangt das Signal zum seriellen Crate-Controller. Das von dort zurückkehrende Signal wird zum Seriellen Highway-Driver zurückgesendet.

entnommen werden. Der AUX-Port muß entweder aktiv über einen CAMAC-IO-Baustein oder passiv über einen Kodierstecker gesteuert werden. Da die aktive Steuerung nur einen geringen Nutzen bringt und für unsere räumlichen Gegebenheiten einen Mehraufwand an Verkabelung erfordert hätte, wurde die zweite Möglichkeit gewählt. Der Kodierstecker legt üblicherweise die drei Pins des Control-Ports, die den Status angeben, auf 0 (GND) und selektiert so die Normalkonfiguration. Es wurde zusätzlich ein Kodierstecker gebaut, in den ein Dip-Schalterfeld integriert ist, so daß sich die entsprechenden Konfigurationen per Hand einstellen lassen. Die Grundkonfiguration wird durch Setzen der Schalter 4-6 auf 'on' gewählt.

Die Steuerung der Konfiguration des Slave-UPAs (Tab. 4.3) erfolgt automatisch über Signale vom SCC und über die beiden an den Eingängen anliegenden Sync-Signale. Zu diesem Zweck werden neben Clock und Bit auch das Loop-Collapse und Bypass-Control-Signal vom SCC zum UPA-Slave geführt.

| Loop Collapse | Sync Main | Sync Backup | Verwendeter In-Loop | Verwendeter Out-Loop |
|---|---|---|---|---|
| On | On | On | no change | |
| On | On | Off | Backup | Backup |
| On | Off | On | Main | Main |
| On | Off | Off | no change | |
| Off | On | On | no change | |
| Off | On | Off | Backup | Main |
| Off | Off | On | Main | Backup |
| Off | Off | Off | no change | |

Tabelle 4.3: Signalweg-Konfigurationen des Slave-UPA. On=0, Off=1. On/On/On ist die Default-Konfiguration

Das Bypassed-Signal wird durch das Bypassed-Bit im Statusregister des SCC gesteuert, welches bei der Initialisierung des Systems zurückgesetzt wird. Es bewirkt die Auskopplung des SCCs aus dem Serial Highway durch Überbrückung des entsprechenden Anschlußports. Da auf diese Weise der SCC als potentielle Störungsquelle eliminiert wird, können Störungen im Serial Highway leichter lokalisiert werden. Der SCC empfängt zwar das Datensignal, reagiert jedoch lediglich auf Befehle zur Änderung seines Status, z.B. zur Aufhebung des Bypassed-Modus. CAMAC-Befehle an die Module des Crates werden ignoriert.

Das Loop-Collapse-Signal beeinflußt zusammen mit den intern im UPA erzeugten SYNC-Signalen die aktuelle Signalweg-Konfiguration. Es wird ebenfalls durch ein Bit im Statusregister des SCC gesteuert. [1] Die verschiedenen Signalweg-Konfigurationen dienen dem Erhalt der Übertragungsfähigkeit der Lichtleiterstrecke auch bei Ausfall eines der Lichtleiter. Im Normalbetrieb wird vom UPA entweder der vollständige Main- oder der Backup-Loop benutzt, je nach Anliegen des Main- bzw. Backup-Sync-Signals. Bei Anliegen des Loop-Collapsed-Signals schaltet der UPA je nach vorhandenem Sync-Signals entweder in eine Konfiguration, in der das Main-Signal als Eingangssignal benutzt wird und das Ausgangssignal über den Backup-Loop zurückläuft oder umgekehrt.

[1] Das Bit kann im Steuerprogramm mit Hilfe des Kommandozeilenbefehls 'loopc' gesetzt werden.

#### Serial Highway-Driver

Der SHD besteht aus einem CAMAC-Modul doppelter Breite und wird wie ein normales CAMAC-Modul über dessen Crate- und Moduladresse mittels Read- und Write-Befehlen gesteuert. Die Befehle, die der SHD auf den Serial Highway übertragen soll, werden dabei in kodierter Form als Parameter übergeben.

Die Konfiguration erfolgt über Jumper und Dip-Schalter im Innern des Moduls. Eingestellt werden Datenfrequenz, Bit/Byte-Serial-Mode und die Anzahl der "Space Bytes", d.h. Pausen-Bytes, die eingebunden werden, um dem angesprochenen SCC Zeit für eine interne Verarbeitung des Kommandos zu geben. Die Anzahl der nötigen Space-Bytes hängt von der für den seriellen Highway gewählten Übertragungsfrequenz ab und ist in der Betriebsanleitung des SHD [Hy01] tabelliert.

#### Serial Crate-Controller

Der SCC benötigt außer der Crate-Adresse, die auf der Frontplatte selektiert werden kann, keine weitere Konfiguration. Eine wichtige Eigenschaft des Controllers ist, daß eine Initialisierung und Selektierung des Non-Bypassed-Modus rein softwaremäßig erfolgen kann. Das anfangs geschilderte Problem des Ausfalls der Steuerung durch Störung des Crate-Controllers entfällt. Bei Fehlfunktion des Crate-Controllers läßt sich dieser einfach über den seriellen Highway in den ursprünglichen Zustand versetzen, ohne daß die Quelle betreten werden muß. Da der UPA-Slave über den SCC gesteuert wird, können auch mit diesem Baustein keine Probleme auftreten.

### 4.3.2 Software-Ansteuerung

#### Initialisierung

**SHD:** Zur Initialisierung wird vom Steuerprogramm zunächst der Status des SHD durch den CAMAC-Befehl F11A12 zurückgesetzt. Dieser löscht das 12-Bit LAM Status-Register, dessen Zustand an den LEDs auf der Frontseite angezeigt wird. Nach der Initialisierung sollte nur noch die SYNC-LED leuchten, ansonsten ist ein Fehler vorhanden. Ausgenommen ist die DMD-LED, die ggf. das Vorliegen eines LAM-Signals anzeigt.

**SCC:** Der SCC wird initialisiert, indem Bit 12 und 13 mit dem CAMAC-Befehl N30F17-A0 ( mit entspr. Crate-Adresse) im Status-Register gelöscht werden. Bit 12=0 hebt den Bypass-Mode des SCC auf, Bit 13=0 setzt den SCC On-Line.

#### Übermittlung von CAMAC-Befehlen mit dem SHD

Der SHD besitzt mehrere Register, u.a. ein Befehls-Register, ein Read-Data und ein Write-Data-Register. Soll ein Befehl auf dem Serial Highway übertragen werden, muß vom Programm der SHD als normaler CAMAC-Baustein angesprochen werden und der kodierte Befehl (incl. CAMAC-Zieladresse) ins Befehlsregister und das zu übertragende Datum ins Datenregister mittels eines CAMAC-Write-Befehls geschrieben werden. Die Übertragung auf dem Serial Highway wird vom SHD durchgeführt, wenn entweder ein Read- oder Funktionsbefehl (also ein Befehl ohne zusätzliche Parameter) ins Befehlsregister geschrieben, oder wenn das Write-Data-Register beschrieben wird, falls sich im Befehlsregister noch ein Write-Befehl befindet. Das Ergebnis eines seriellen Read-Befehls liegt nach erfolgreicher Übertragung im Read-Data-Register des SHD.

Die erfolgreiche Ausführung des Befehls, insbesondere das Eintreffen einer korrekten Rückantwort bei einem Read-Befehl, wird durch Setzen des RDY-Bit im Statusregister angezeigt. Dieses läßt sich durch eine spezielle Testfunktion des SHD, die ein Q-Signal zurückgibt, prüfen. Die steuernde Software wartet nach Absendung eines Read-Befehls in einer Schleife, bis das RDY-Bit gesetzt ist, und liest dann erst das Read-Data-Register aus, da ansonsten wegen der durch die serielle Übertragung auftretenden Zeitverzögerung noch das Ergebnis eines alten Read-Befehls gelesen würde.

Das Auftreten eines Q- oder X- Signals auf dem seriellen Highway kann ebenfalls über eine spezielle Funktion des SHD abgefragt werden.

## 4.4 Quellencrate

Die Elektronik auf der Quelle wurde von der alten Steuerung übernommen. Zum Umbau zwischen alter und neuer Steuerung wurde lediglich der Crate-Controller gegen den neuen seriellen Controller ausgetauscht.

Das Crate auf der Quelle enthält neben dem seriellen Crate-Controller folgende Bausteine:

**Position 1-8** Acht doppelte 16-Bit Ausgaberegister 20R2008. Diese speichern die digitalen Sollwerte. Über DAC-Platinen in der Plattformzentrale werden diese mit einer Auflösung von 12 Bit in Spannungen umgewandelt, mit denen die Netzgeräte gesteuert werden. Diese DAC-Platinen enthalten ebenfalls die Vorrichtung für die automatische Werteübernahme, die mit dem Setzen von Bit 16 im Register angefordert wird.

Einige der neuen Netzgeräte besitzen diesen Werteübernahmemechanismus nicht oder werden binär bzw. BCD[2]-kodiert angesteuert.

**Position 15** Doppeltes 24-Bit Eingaberegister JRE10. Dieses empfängt die digitalen Istwerte von der Plattformzentrale und hält sie für die CAMAC-Übertragung bereit. Register 0 enthält die Informationen der digitalen Parameter wie Spinstellung und Strahlrichtung. Register 1 enthält den Extern/Intern- Zustand der Netzgeräte.

**Position 16** Doppeltes 24-Bit Ausgaberegister JRS10. Es enthält die digitalen Sollwerte für Spinstellung und Strahlrichtung.

**Position18** Multiplexer 1701. Der Multiplexer wählt einen aus 16 Kanälen, die die Rückmeldespannung der Netzgeräte liefern, für die Umwandlung mit dem ADC aus.

**Position 19 u. 20** ADC 1241. Der ADC dient zur digitalen Umwandlung der Rückmeldespannung.

[2]Binary Coded Decimal

# Kapitel 5

# Umbau des Kontrollpultes

Um eine möglichst geringe Umstellung für den Benutzer beim Übergang auf das neue Fernsteuersystem zu erreichen, und da sich die Bedienung der einzelnen Kanäle über Potentiometer als sehr zweckmäßig zur Einstellung des Strahls erwiesen hat, wurde das Kontrollpult der bestehenden Steuerung in das neue Fernsteuerungssystem integriert. Hierbei wurden lediglich die Treiberplatinen und die Fernbedienungseinschübe mit den Anzeige- und Bedienungselementen beibehalten, die Steuerplatinen jedoch entfernt. Deren Aufgabe wird nun durch den Rechner übernommen.

## 5.1 Ansteuerung der Treiberplatinen

Die Treiberplatinen bilden eine Schnittstelle zwischen Anzeigeinstrumenten bzw. Bedienungselementen und dem internen Datenbus des Steuerpults. Diese Platinen haben Aufgaben der Digitalisierung, Pegelwandlung, Zwischenwertspeicherung und Selektierung der entsprechenden Signale (s. Abb. 5.1). Schaltpläne der Platinen finden sich in [An76].

Die wesentlichen Signale, die für die Steuerung benötigt werden, sind der 16-Bit Read-Bus R, von dem die Platinen Daten empfangen, und der 16-Bit Write-Bus W, an dem die Datensignale der Platinen anliegen. Desweiteren existieren die Adressleitungen A-E, auf denen die aktuelle Kanalnummer des Multiplexers anliegt, insgesamt 6 Strobe- und Enable-Signale für die veschiedenen Platinen, und das End-of-Conversion-Signal (EOC) des ADCs. Da der manuelle Branch-Driver für die neue Steuerung nicht mehr benutzt wird, wurden die beiden 50-poligen Sub-D-Stecker, die zu dessen Anschluß dienten, zur Verbindung der Fernsteuerzentrale mit dem Ein/Ausgaberegister der neuen Steuerung kompatibel erweitert. Die R- und W- Signale lagen an den beiden Steckern bereits an, die fehlenden Signale wurden an freie Pins dieser Stecker geführt. Dadurch war nur eine geringfügige Zusatzverkabelung nötig, die überdies den Umbau in den alten Zustand durch einfaches Umstecken der Kabel ermöglicht.

### 5.1.1 Analog-Anzeige

Die Platine AAV übernimmt auf Signal Z3 den Wert der Read-Leitungen R1-R4 in ein Latch. Der Ausgang dieses Latches liegt am Eingang eines 16 aus 4-Decoders, der auf Signal Z4 hin den durch die 4-Bit-Zahl am Eingang bestimmten Ausgang auf LO schaltet. Jeder dieser 16 Ausgänge bildet das Strobe-Signal für jeweils einen Fernbedienungseinschub. Dieser übernimmt auf das Strobe-Signal hin die acht höherwertigen Bits des R-Busses (R4-R11) in ein Latch und steuert mit diesem Wert einen DAC an, dessen Ausgangsspannung ein

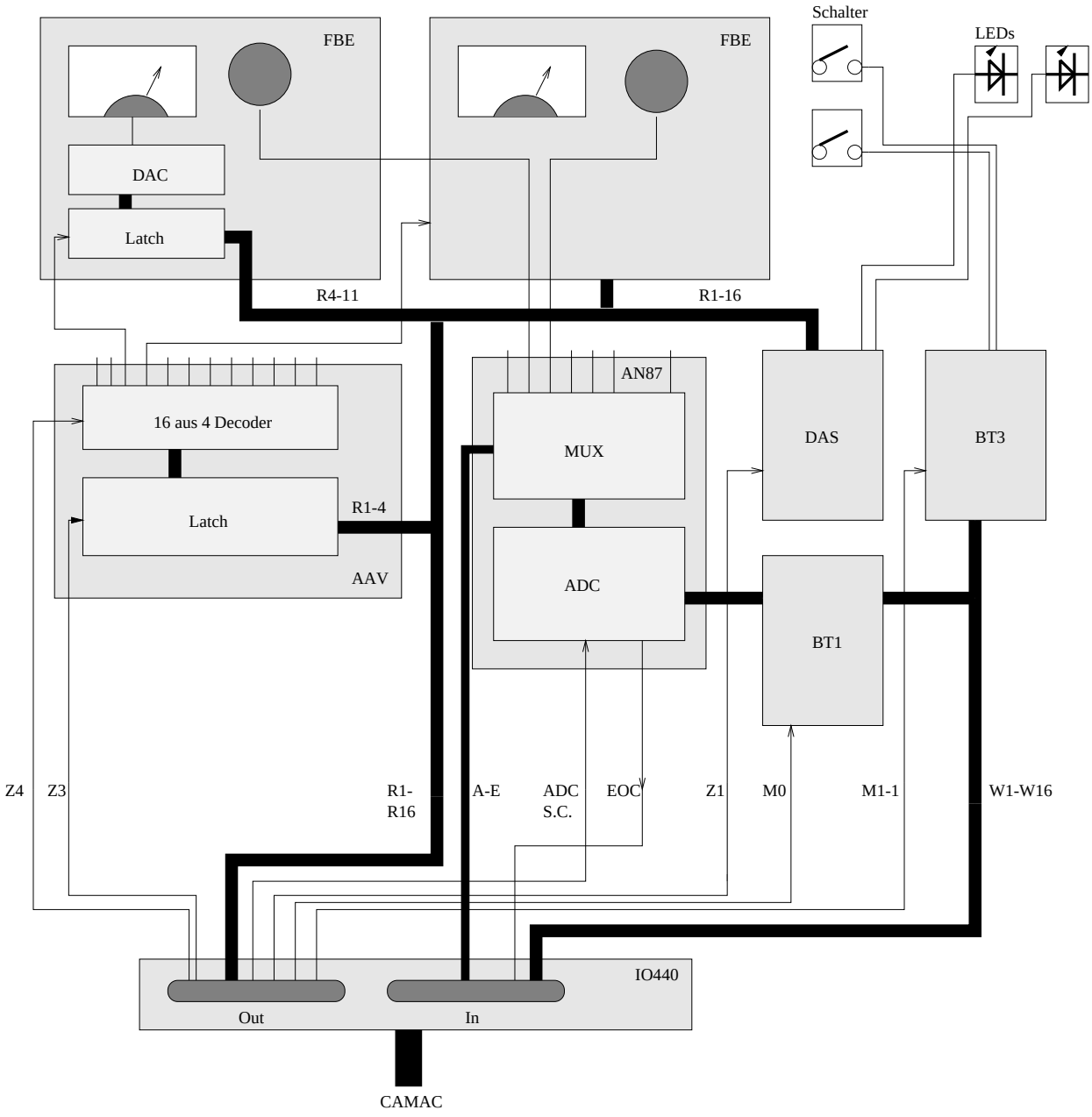


Abbildung 5.1: Treiberplatinen und Signale des Kontrollpultes

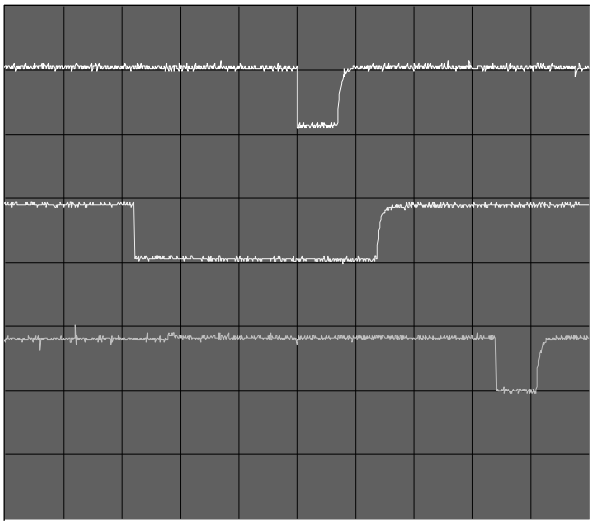

Abbildung 5.2: Signale der AAV-Platine zur Ansteuerung der Analoganzeige.
X-Achse: 1 Einheit = $50\mu sec$. Y-Achse: 1 Einheit = $5V$.
Von oben nach unten: Strobe-Signal Z3, Datensignal R5, Strobesignal Z4

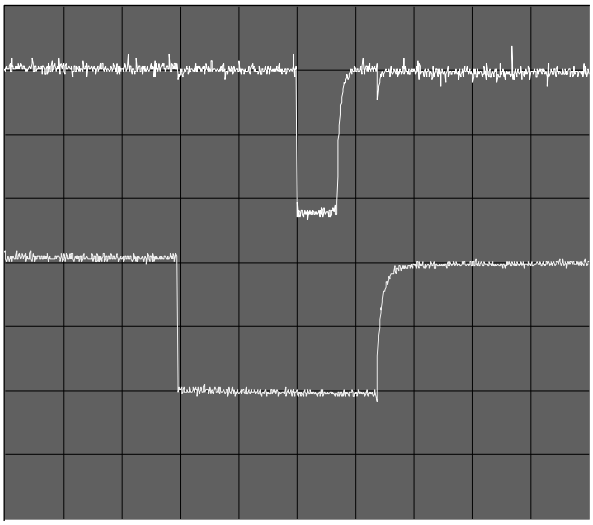

Abbildung 5.3: Signale der DAS-Platine zur Ansteuerung der Anzeige-LEDs.
X-Achse: 1 Einheit = $50\mu sec$. Y-Achse: 1 Einheit = $2V$.
Oben: Strobe-Signal Z1. Unten: Datensignal R5.

Drehspulmeßinstrument treibt. Der zeitliche Ablauf der Signale ist in Abb. 5.2 dokumentiert.

Der 16-aus-4 Decoder-IC 74154 der Platine besitzt zwei logisch UND-verknüpfte Strobe-Eingänge, die beide auf LO liegen müssen, um den IC zu aktivieren. Die benötigten Strobe-Signale wurden im Aniol-System durch die Signale Z4 und R12+R13 realisiert. Das Signal R12+R13 diente in der Aniol-Steuerung internen Synchronisationszwecken und ist in der neuen Steuerung nicht mehr notwendig. Die entsprechenden Signalleitungen wurden deshalb innerhalb des Anschlußsteckers miteinander verbunden, so daß beide Signale nur 1 Bit des ansteuernden CAMAC-IO-Registers benötigen. Der bei Rückbau der Steuerung in den alten Zustand erfolgende Austausch der Anschlußstecker sorgt für einen automatischen Wegfall der Verbindung und erhält so die Kompatibilität zum alten Fernsteuerungssystem.

### 5.1.2 Digital-Anzeige (LEDs)

Die Platine DAS übernimmt auf das Signal Z1 hin die Werte R1-R16 und speichert diese in einem Latch (s. Abb. 5.3). Dessen Ausgänge werden bitweise zur Ansteuerung der verschiedenen LEDs benutzt.

### 5.1.3 Digital-Schalter

Während des Enable-Signals M1-1 schaltet die Platine BT3 die Kippschalter für Polarisation, Quench all und Cup/Tandem durch und legt diese auf die Write-Leitungen W1-W16 (s. Abb. 5.4).

### 5.1.4 Analog-Potentiometer

Die Platine AN87 enthält einen MP6912 ADC/Multiplexer-Baustein der Firma Analogic. Dieser ist so konfiguriert, daß bei jedem Start-Conversion-Signal die Kanalnummer zyklisch erhöht wird. Der ADC wandelt die durch das Potentiometer im entsprechenden Fernbedienungseinschub gegebene Spannung in einen digitalen Wert. Nach erfolgreicher Konversion der Daten sendet der ADC ein EOC-Signal. Während des Signals M0 schaltet die Platine BT2 den digitalen Ausgangswert des ADCs auf die Leitungen W1-W12 (s. Abb. 5.5). Zu beachten ist, daß diese in bitweise gespiegelter Form vorliegen, d.h. das Most-Significant-Bit ist Bit 0.

Die Nummer des gerade konvertierten Kanals ist auf die ursprünglichen Eingänge der Platine für die Kanalnummer gelegt und kann über die Leitungen A-E gelesen werden.

### 5.1.5 Extern/Intern-LEDs

Die einzelnen Fernbedienungseinschübe enthalten LEDs, die den aktuellen Status der Netzgeräte (interne oder externe Steuerung) anzeigen. Die Treiber dieser LEDs befinden sich in den Fernbedienungseinschüben. Die Werte der entsprechenden Bits der Read-Leitungen werden auf das Strobe-Signal Z5 hin übernommen.

## 5.2 Anbindung an das CAMAC-System

Die Ansteuerung der Treiber-Platinen erfolgt über das 2x24-Bit-I/O-Register 440 der Firma Hytec [Hy04]. Dabei werden über den Out-Port die Strobe- bzw. Enable-Signale, das Start-Conversion-Signal und die Werte für die R-Leitungen geschrieben. Über den In-Port werden

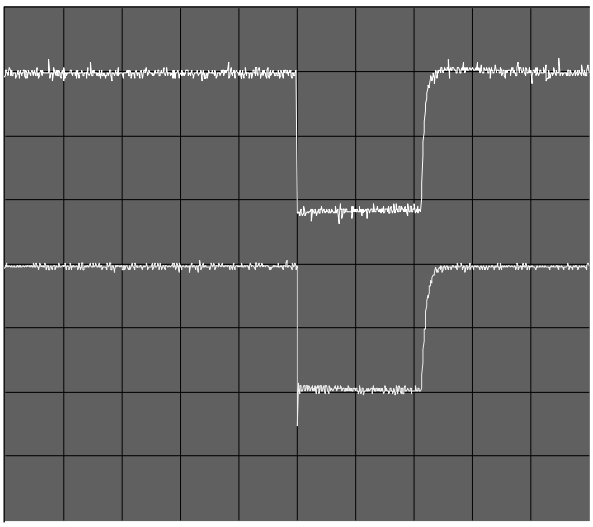

Abbildung 5.4: Signale der BT3-Platine zur Abfrage der Digitalschalter.
X-Achse: 1 Einheit = $50\mu sec$. Y-Achse: 1 Einheit = $2V$.
Oben: Enable-Signal M1-1. Unten: Daten-Antwortsignal W5.

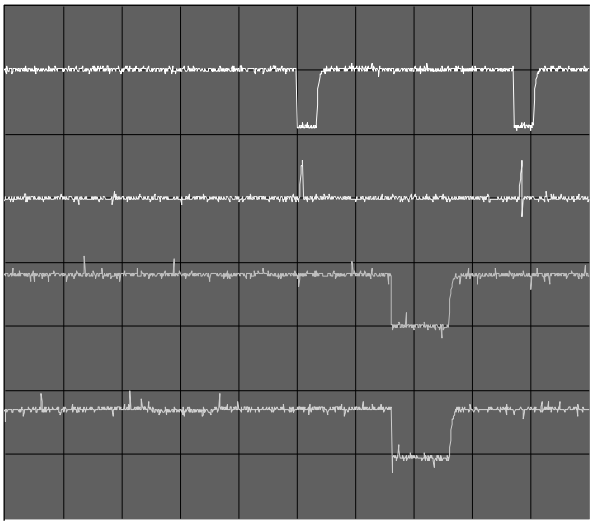

Abbildung 5.5: Signale der ADC- und BT1-Platine zum Auslesen der Potentiometer-Werte.
X-Achse: 1 Einheit = $100\mu sec$. Y-Achse: 1 Einheit = $5V$.
Die Signale von oben nach unten: ADC Start Conversion, ADC End Of Conversion, BT1-Strobesignal M0, Datensignal W3. Wie man durch Vergleich des Start-Conversion- und des End-Of-Conversion-Signals sieht, arbeitet der Pult-ADC deutlich schneller als das PC/CAMAC-System. Die Konversion ist bereits beendet, bevor das Start-Conversion-Signal abgeschlossen ist.

die Leitungen W1-W16, Die Kanalnummern A-D und das EOC-Signal gelesen. Auf die Ansteuerung des Registers durch die Software wird in Abschnitt 8.2.2 eingegangen.

Das Input-Register ist über eine Drahtbrücke so konfiguriert, daß der Wert des Registers vor jedem CAMAC-Read-Kommando neu vom Eingabe-Port gelesen wird und kein externes Strobe-Signal benötigt.

Das Output-Register besitzt Open-Collector-Ausgänge. Um einen definierten Spannungszustand zu erreichen, sind daher an allen Ausgängen über einen Zwischenstecker Pull-Up-Widerstände von 1 $k\Omega$ gegen +5V geschaltet.

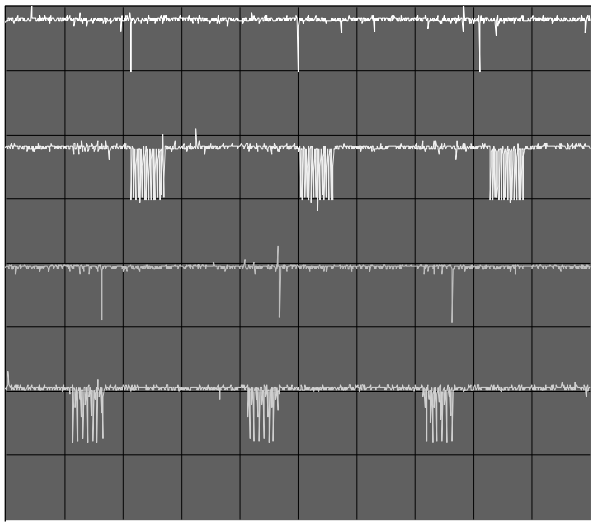

Abbildung 5.6: Dieser Graph dokumentiert das Timing des kompletten Steuerungszyklus. X-Achse: 1 Einheit = $10\mu sec$. Y-Achse: 1 Einheit = $5V$.
Signale von oben nach unten: M1-1 (Enable-Signal der Digitalschalter-Treiberplatine BT3), M0 (Enable-Signal der Treiberplatine für den ADC-Wert), Z1 (Strobe-Signal zur Übernahme der Digitalwerte), Z3 (Strobe-Signal zur Übernahme der Analog-Kanäle).

# Teil III

# Software

# Kapitel 6

# Überblick über die Software

## 6.1 Problemstellungen

Einen entscheidenden Teil der vorliegenden Arbeit nahm die Entwicklung geeigneter Steuerungssoftware ein. Um den gestellten Anforderungen zu genügen, waren folgende Aufgaben zu lösen:

1. Auswahl und Installation eines geeigneten Betriebssystems.
2. Entwicklung von Treiberroutinen für das CAMAC-Interface.
3. Ermöglichung der Steuerung des Programmes über die Terminal-Tastatur mit unmittelbarer Reaktion auf Tastendruck des Benutzers.
4. Integration des Kontrollpultes der alten Steuerung als alternative Bedienungsmöglichkeit des Programmes. Softwaremäßige Emulation der Funktion der Steuerplatinen.
5. Weitgehende Reduzierung der Rechenzeit, um den Ablauf anderer Programme nicht zu beeinträchtigen.
6. Schaffung einer Möglichkeit zur Kommunikation mit anderen Programmen (IPC[1]).
7. Einbindung einer grafischen Benutzeroberfläche zur Steuerung des Programmes.
8. Berücksichtigung von möglichen Erweiterungen der Steuerung.
9. Implementierung von Routinen zur Protokollierung und Speicherung der Quellenparameter.
10. Automatisierung der Übernahme der auf der Quelle eingestellten Werte auf die Fernsteuerung.
11. Sicherstellen der periodischen Auffrischung der Quellenwerte und der Werteanzeigen.
12. Maßnahmen zur Fehlererkennung und automatischen Überwachung der Quellenparameter.

[1] Inter Process Communication

## 6.2 Problemlösungen

1. Wegen seiner Multitasking-Fähigkeit, Netzwerkunterstützung und Kompatibilität zu UNIX wurde auf dem Steuer-PC das Betriebssystem LINUX installiert.

2. Unter LINUX wurden Treiberroutinen zur Steuerung der CAMAC-Interface-Karte entwickelt.

3. Durch Nutzung der Tastatur-*Signals* [2] und des direkten Zugriffs auf den Terminal-Driver ermöglicht das Programm die komfortable Steuerung der Quelle auch von einfachen Text-Terminals und Terminal-Emulationen aus.

4. Die Bedienelemente des Steuerpultes wurden über ein CAMAC-Ein/Ausgaberegister in die neue Steuerung integriert. Für die Ansteuerung der Elemente wurden Treiberroutinen entwickelt.

5. Durch Verwendung der UNIX-*Signal*-Technik (s. Abschn. 7.4) reagiert das Programm unmittelbar auf äußere Events wie Tastendruck des Benutzers oder Signale des Systemtimers, ohne daß eine rechenzeitintensive kontinuierliche Abfrage der entsprechenden Komponenten erfolgen muß.

6. Eine nach dem Shared-Memory-Prinzip funktionierende Software-Schnittstelle ermöglicht das Senden von Steuerbefehlen aus externen Programmen an das Steuerprogramm.

7. Die Programmiersprache Tcl/Tk hat sich wegen ihrer Effektivität zur Entwicklung grafischer Oberflächen unter UNIX durchgesetzt. Im Rahmen dieser Arbeit wurde ein erweiterter Tcl/Tk-Interpreter erstellt, der über das Software-Interface auf die Quellensteuerung zugreifen kann. Mit Hilfe dieses Interpreters können schnell und einfach grafische Benutzeroberflächen den Anwenderwünschen entsprechend entwickelt werden. Eine erste Oberfläche wurde im Rahmen dieser Arbeit erstellt.

8. Umfangreiche Konfigurationsmöglichkeiten, die Möglichkeit des Zugriffs anderer Programme auf das Steuerprogramm und eine starke Strukturierung des Quelltextes ermöglichen die leichte Erweiterbarkeit der Steuerung.

9. Routinen zum Speichern und Laden des aktuellen Quellenzustandes wurden entwickelt. Ebenso wurden Möglichkeiten geschaffen, die Daten der Quelle zu protokollieren. Das Format ist so gewählt, daß es mit dem gebräuchlichen Programm AWK bearbeitet werden kann. AWK ist ein Skriptspracheninterpreter zur zeilenorientierten Textverarbeitung, der u.a. zur Extraktion von Datensätzen aus ASCII-Protokollfiles besonders geeignet ist. Näheres findet sich in Abschnitt 10.5.

10. Die Werteübernahme wurde, soweit es die Hardware zuließ, automatisiert.

11. Die periodische Auffrischung der Quellenwerte erfolgt über einen vom Systemtimer ausgelösten Interrupt.

12. Vorrichtungen zur Behandlung von Übertragungsfehlern und Alarmeinrichtungen zur Anzeige von Fehlern und Schwellwertüberschreitungen auf der Quelle wurden implementiert.

[2] *Signal* ist unter UNIX ein feststehender Begriff. Um Verwechslungen mit anderen Signalen zu vermeiden, sind UNIX-*Signals* im folgenden Text kursiv gesetzt.

Zusätzlich wurden einige nützliche Funktionen, wie die Untersetzung der Potentiometer oder das explizite Senden von CAMAC-Kommandos, eingebaut.

# Kapitel 7

# Verwendete Techniken und Verfahren

Für die Erstellung der Software waren eine Reihe von speziellen Softwarelösungen und Programmiertechniken nötig, die in diesem Kapitel getrennt erläutert werden. Auf den Aufbau des Steuerprogrammes wird im nächsten Kapitel eingegangen.

## 7.1 Das Betriebssystem LINUX

Für die Entwicklung der Steuerung ist die Wahl des Betriebssystems von entscheidender Bedeutung. Um Aufgaben wie die Einbindung externer Programme zur Erzeugung grafischer Benutzeroberflächen oder Überwachung der Quelle erfüllen zu können, muß das Betriebssystem die Bearbeitung mehrerer Prozesse nebeneinander und eine Übertragung von Daten zwischen diesen Prozessen ermöglichen. Für die Steuerung und Überwachung der Quelle von externen Terminals aus ist eine problemlose Anbindung des Rechners ans Institutsnetz nötig. Alle diese Voraussetzungen werden durch UNIX-kompatible Betriebssysteme erfüllt.

Aufgrund der technischen Gegebenheiten (Vorhandensein zweier SPARC-Stations unter LynxOS, für die bereits CAMAC-Treiberroutinen existierten [Ni94]) wurde das Steuerprogramm zunächst unter dem UNIX-kompatiblen LynxOS entwickelt. Diese Rechner standen jedoch nach einiger Zeit nicht mehr zur Vefügung, so daß die langfristig geplante Bereitstellung eines eigenen PCs für die Quellensteuerung vorgezogen wurde. Wegen seiner UNIX-Kompatibilität wurde das Betriebssystem LINUX installiert.

LINUX ist ein vollwertiges UNIX (POSIX) - kompatibles Betriebssystem für IBM-kompatible PCs, welches Ende 1991 entwickelt wurde und kostenlos erhältlich ist. Das System enthält einen C-Compiler, die Interpretersprache Tcl/Tk, die grafische Benutzeroberfläche X-Windows und verschiedene gebräuchliche Dienstprogramme wie AWK und Gnuplot.

Für das vorliegende Projekt war LINUX aus folgenden Gründen besonders geeignet:

- Wegen der beiderseitigen UNIX-Kompatibilität war eine Umstellung des Programmes von LynxOS auf LINUX problemlos möglich. Kleine Anpassungen waren lediglich bei den sehr systemnahen Funktionen, z.B. zur Behandlung der UNIX-*Signals*, nötig. Außerdem mußten Treiberroutinen für die Ansteuerung der CAMAC-Karte entwickelt werden.

- LINUX ist multiuser- und multitaskingfähig. Verschiedene Programme können gleichzeitig auf dem Rechner genutzt werden, ebenso können verschiedene Benutzer zur selben Zeit den Rechner benutzen.

- LINUX bietet die unter UNIX bekannten Möglichkeiten zur Interprozesskommunikation, d.h. dem Austausch von Daten zwischen verschiedenen parallel laufenden Prozessen, und dem *Signal*-Handling, d.h. der Reaktion eines Prozesses auf Signale des Betriebssystems, die z.B. bei Betätigung der Tastatur oder Ablauf eines internen Timers generiert werden.

- Ein PC unter LINUX kann aufgrund seiner UNIX-Kompatibilität und der integrierten Netzwerkunterstützung unter TCP/IP [1] problemlos am Institutsnetz betrieben werden.

- Da LINUX kostenlos erhältlich ist, ist es unter allen Betriebssystemen die kostengünstigste Lösung.

- Unter LINUX sind eine Reihe wichtiger Programme bereits vorhanden, wie ein C-Entwicklungssystem, ein Tcl/Tk-Interpreter, X-Windows, Gnuplot u.v.m.

- LINUX ist ein 32-Bit-System, hat also gegenüber z.B. MS-DOS einen entscheidenden Geschwindigkeitsvorteil. Gegenüber anderen 32-Bit-Betriebssystemen wie Windows 95 oder OS/2 besitzt es die genannten Vorzüge der UNIX-Kompatibilität.

- Neben dem mit der 32-Bit-Technik verbundenen größeren Adreßbereich, der eine im Gegensatz zu DOS wesentlich effektivere Speicherverwaltung ermöglicht, benutzt LINUX zusätzlich eine Partition der Festplatte als virtuellen Arbeitsspeicher und ist so auch zur Bearbeitung sehr speicherplatzintensiver Programme in der Lage.

## 7.2 LINUX-Gerätetreiber für die CAMAC-Karte

Weil es sich bei LINUX um ein recht neues und dazu nichtkommerzielles Betriebssystem handelt, standen fertige Treiberroutinen für die CAMAC-Interface-Karte nicht zur Verfügung, so daß diese im Rahmen der Arbeit entwickelt werden mußten. Da das neue System auf PC-Basis wegen des Wegfalls der LynxOS-Rechner möglichst schnell zur Verfügung stehen mußte, wurde auf die Programmierung eines ins Betriebssystem eingebundenen UNIX-Gerätetreibers für die CAMAC-Karte zunächst verzichtet, da dieses einen erheblichen Mehraufwand dargestellt hätte. Stattdessen wurden die Treiberroutinen direkt ins Steuerprogramm integriert.

Mit der CAMAC-Karte des LINUX-PCs ergab sich eine Übertragungsrate von 30000 CAMAC-Befehlen/sec im Gegensatz zu 1000 Befehlen/sec unter dem alten LynxOS-System.

### 7.2.1 Programmierung der CAMAC-Karte

In IBM-kompatiblen PCs erfolgt der Zugriff auf Hardware-Steckkarten über Ein-/Ausgabeports, denen Adressen im Speicherbereich des Rechners entsprechen. Die CAMAC-Karte der Firma Wiener wird über drei jeweils 2 Bytes breite IO-Ports angesprochen.

Das Schreiben des ersten Ports dient zum Setzen des Primary-Mode, der bestimmt, welche Bedeutung der dem zweiten Port übergebene Wert haben soll. Primary-Mode PM1 bedeutet, daß mit Schreiben von Port 2 ein Sub-Mode gesetzt wird. PM2 bedeutet das Schreiben eines Datenwortes und PM3 das Lesen. Der gesetzte Sub-Mode wiederum bestimmt die Bedeutung des nächsten Datenwortes. Sub-Mode SM2 z.B. gibt an, daß das

[1] Transfer Control Protocol / Internet Protocol

nächste Wort als die unteren 16 Bit des Datensignals auf den CAMAC Dataway geschrieben werden soll, Sub-Mode SM1 gilt entsprechend für die restlichen oberen 8 Bit.

```
set PM1       - Setzen Primary-Mode 1: Setzen Submode
set SM0       - Setzen Submode 0: Das naechste Wort ist der CAMAC-Befehl
set PM2       - Setzen Primary-Mode 2: Uebertragung Daten
write NAF     - Sende CAMAC-Befehls/Adresswort
set PM1       - Setzen Primary-Mode 1: Setzen Submode
set SM2       - Setzen Submode 0: Das naechste Wort ist das Low-Word
set PM2       - Setzen Primary-Mode 2: Uebertragung Daten
write W1-W16 - Sende Datenwort
```

Abbildung 7.1: Beispiel für die Übertragung eines 16-Bit-Wortes über das CAMAC/PC-Interface. (Pseudo-Code)

Auf diese Weise kann mit Hilfe einer Sequenz von 8-10 Port-Befehlen ein CAMAC-Befehl an den Crate-Controller gesendet werden.

Bei der Initialisierung des CAMAC-Interfacesystems wird zunächst die Crate-Nummer des anzusprechenden Crate-Controllers festgelegt. Alle weiteren CAMAC-Befehle beziehen sich auf diese Crate-Nummer, so daß der Interfacekarte mit jedem CAMAC-Befehl lediglich ein zusammengesetztes Wort aus N, A und F zur Adressierung des Befehls übergeben wird.

Die implementierten Befehle sind kompatibel zur am Institut entwickelten CAMAC-Befehlsbibliothek für LynxOS von Norbert Nicolay [Ni94], so daß das Steuerprogramm ohne Probleme von LynxOS auf LINUX umgestellt werden konnte.

### 7.2.2 Ansprechen der IO-Ports unter LINUX

Um einzelne IO-Ports unter LINUX direkt ansprechen zu können, müssen diese zunächst vom System freigegeben werden. Die Anforderung dieser Freigabe geschieht durch die C-Funktion *ioperm (port, num, 1)*, der die Adresse, die Anzahl der Bytes und ein Flag, welches den gewünschten Status (freigeben oder sperren) angibt, übergeben werden. Für die Freigabe des Ports ist es erforderlich, daß das Programm unter Superuser-Zugriffsrechten ausgeführt wird.

Nach erfolgter Freigabe können Werte auf die einzelnen Ports mit den im LINUX-Kernel implementierten Routinen *outw* bzw. *outb* (für Word- bzw- Byte-Länge) geschrieben und mit *inb* bzw. *inw* gelesen werden. Diese Befehle sind in der Include-Datei ⟨ *asm/io.h* ⟩ definiert. Abschließend wird der Port mit *ioperm(port,num,0)* wieder gesperrt.

Routinen, die das obige Verfahren benutzen, müssen mit der Optimierungsoption *-O2* und ohne die Debugging-Option *-g* compiliert werden, da es sonst zu Timingfehlern kommen kann. [Be94, Li01]

## 7.3 Tcl/Tk

Die Bedienung eines Programmes über grafische Benutzeroberflächen ist von großem Vorteil, da sie meist intuitiv erfolgen kann und keine große Einarbeitungszeit erfordert. Die Programmierung einer solchen Oberfläche in C ist jedoch sehr aufwendig. Ein Schritt zur

```
#include <asm/io.h>   /* Definition outw-Befehl */
...
#define BADR0 0x380   /* IO-Adressen fuer PC16-Karte */
#define BADR2 0x382
#define BADR4 0x384
...
void wr_cam(CAMADR *adr, int a, int f, UINT16 *v)
{
  int nfa;
  nfa=((a<<5)|f)<<5;      /* Errechnung nfa-Word */
  nfa|=(adr->station);    /* aus CAMAC-Adresse und Subadresse */
  outw(1,BADR0);          /* set PM1    */
  outw(0,BADR2);          /* set SM0    */
  outw(2,BADR0);          /* set PM2    */
  outw(nfa,BADR2);        /* write nfa  */
  outw(1,BADR0);          /* set PM1    */
  outw(2,BADR2);          /* set SM2    */
  outw(2,BADR0);          /* set PM2    */
  outw(*v,BADR2);         /* write data */
}
```

Abbildung 7.2: Implementierung des Beispieles aus 7.1 in C

Vereinfachung dieser Aufgabe ist die unter den meisten UNIX-Systemen verfügbare Interpretersprache Tcl und das dazugehörige Grafik-Tool Tk.

Die Programmierung eines einfachen Kommando-Buttons auf dem Bildschirm erfordert bei direkter C-Programmierung mehrere Seiten Quelltext. Die Verwendung eines Widget-Paketes wie Motif oder Athena stellt schon eine erhebliche Verbesserung dar, erfordert aber dennoch ein ca. 50 zeiliges Programm. In Tcl/Tk erfordert die Programmierung lediglich 2 Zeilen.

Ein weiterer Vorteil von Tcl/Tk ist die komfortable Schnittstelle zu C und die damit verbundene einfache Erweiterbarkeit durch eigene Funktionen. Dies macht Tcl/Tk zu einem hervorragenden Werkzeug zur Erstellung grafischer Oberflächen für C-Programme.

### 7.3.1 Konzept

Tcl/Tk ist eine Interpretersprache, d.h. die Quelltexte können, ohne kompiliert werden zu müssen, direkt ausgeführt werden. Das macht Tcl/Tk-Programme auch besonders geeignet für Modifikationen durch den Benutzer. Es ist problemlos möglich, den Interpreter um eigene (in C implementierte) Kommandos zu erweitern, die dann wie normale Tcl-Kommandos in Tcl/Tk-Programmen benutzt werden können.

Ein Nachteil gegenüber anderen Programmiersprachen ist außer der durch das Interpreterkonzept bedingten geringeren Geschwindigkeit auch die geringe Anzahl der zur Verfügung stehenden Datentypen: Tcl kennt nur den String und das String-Array, komplexere Datenstrukturen müssen daher in C implementiert werden.

Einige nützliche Konzepte von Tcl/Tk sind das Variable-Tracing und das Event-Binding. Das Tracing erlaubt die direkte Reaktion auf die Änderung eines Variablenwertes ohne expliziten Aufruf. So braucht für die Ausgabe eines Variablenwertes in einem Tk-Displayelement

nur einmal die anzuzeigende Variable definiert zu werden, eine Aktualisierung der Ausgabe erfolgt automatisch bei jeder Änderung des Variablenwertes. Beim Event-Binding können verschiedene Aktionen des Benutzers an Prozeduren gekoppelt werden. Wird z.B. mit der Maus ein Kommando-Button in einem Tk-Window gedrückt, wird automatisch die damit verbundene Prozedur aufgerufen, ohne daß das Event explizit abgefragt werden muß.

Die grafischen Oberflächen werden aus sog. Widgets zusammengesetzt. Das sind Grafikelemente, denen Tcl-Routinen als Reaktionen auf äußere Ereignisse wie z.B. Betätigung durch den Mauszeiger zugeordnet werden. Unter Tk sind einige Standard-Widgets definiert, eigene Widgets können in C implementiert werden.

### 7.3.2 Die Schnittstelle zu C

```
#include <stdio.h>
#include <string.h>
#include <tcl/tcl.h>
#include <tcl/tk.h>

int tcl_MyCommand(ClientData clientData, Tcl_Interp *interp,
                  int argc, char *argv[])
                             /* Dies ist das neue Tcl-Kommando */
{
int my_result=42;
Tcl_SetResult(interp,my_result,TCL_STATIC);
                              /* Rueckgabe des Tcl-Ergebnisses */
return TCL_OK;
}

int Tcl_AppInit(Tcl_Interp *interp)  /* Initialisierungprozedur */
{
  Tcl_Init(interp);  /* Initialisierung von Tcl */
  Tk_Init(interp);  /* Initialisierung von Tk */

  Tcl_CreateCommand(interp,"mycommand",tcl_MyCommand,
                    (ClientData *)NULL, (Tcl_CmdDeleteProc *)NULL);
                /* Definition des neuen Tcl-Kommandos 'mycommand' */
                 /* durch die C-Funktion 'Tcl_MyCommand'
  return TCL_OK;
}

int main(int argc, char *argv[])
{
  Tk_Main(argc, argv, Tcl_AppInit);  /* Aufruf des Main-Loops      */
                                     /* Die Init-Prozedur wird als */
                                     /* uebergeben Parameter       */
}
```

Abbildung 7.3: Beispiel für die Implementation einer erweiterten Tcl/Tk-Shell in C

Tcl/Tk ist sehr geeignet für die Einbindung in C-Programme. Der normale Tcl/Tk-Interpreter *Wish* liegt als C-Bibliothek vor und läßt sich so in eigene C-Programme einbinden. Zusätzlich kann er um eigene in C programmierte Befehle erweitert werden, die in Tcl/Tk-Programmen verwendet werden können. Der für die Quellensteuerung entwickelte Interpreter wurde beispielsweise um Tcl-Befehle ergänzt, die Steuerkommandos an das Quellenprogramm schicken oder von der Quelle Statusinformationen erfragen.

Ein selbstprogrammierter Tcl/Tk-Interpreter besteht im wesentlichen aus folgenden Teilen:

- Einer benutzerdefinierten Init-Prozedur, die dazu dient, eigene Erweiterungen in den Interpreter einzubinden
- Dem Main-Programm, das meist nur einen Aufruf der in der Tk-Library definierten Routine *Tk_Main* enthält.

Die selbstdefinierten C-Funktionen müssen einer bestimmten Syntax gehorchen, die in den Dokumentationen zu Tcl/Tk beschrieben wird. Parameter werden in mehreren Strings übergeben, Rückgabewerte werden in einer Datenstruktur , die dem Interpreter zugeordnet ist, zurückgegeben. Die C-Funktionen werden in der Init-Prozedur durch Aufruf der in der Tcl-Library enthaltenen Funktion *Tcl_CreateCommand* einem Tcl-Kommando zugeordnet.

Es besteht weiterhin die Möglichkeit, mittels des Variable-Tracings einer Tcl-Variablen eine C-Variable zuzuordnen. Dabei wird der Wert der zugeordneten Variablen ständig durch den Interpreter überprüft und bei einer Werteänderung der einen Variablen der Wert automatisch in die andere übertragen. Auf diese Weise kann in beiden Programmiersprachen direkt auf die Variable zugegriffen werden. Nachteil ist, daß die C-Variable dynamisch angelegt werden muß, da sie vom Tcl-Interpreter bei jeder Werteänderung freigegeben und neu angelegt wird. Das macht das Verfahren für Shared-Memory-Bereiche ungeeignet, da sich deren Adressbereich wegen des Zugriffs aus mehreren unterschiedlichen Programmen nicht ändern darf. Es gibt jedoch die Möglichkeit, von C aus einer Tcl-Variablen einen Wert zuzuweisen. Von dieser Möglichkeit wurde im Steuerprogramm Gebrauch gemacht.

### 7.3.3 XF

Ein nützliches Hilfsmittel zur Tcl/Tk-Programmierung ist das Programm *XF*. Es bietet die Möglichkeit, grafische Benutzeroberflächen direkt am Bildschirm nach dem "What-You-See-Is-What-You-Get"-Prinzip zu erstellen. Der Vorteil liegt in der direkten Sichtbarkeit des Ergebnisses. Ebenso sind weniger Kenntnisse in der Sprache selbst erforderlich, obgleich man um Grundkenntnisse nicht herumkommt. Der Nachteil liegt in einem recht ineffektiven und unübersichtlichen Code (ca. 10 mal größer als ein vergleichbares manuell programmiertes Programm). Zum Erstellen von einfachen Oberflächen ist dieses Werkzeug jedoch durchaus geeignet.

## 7.4 Signals

Für das Steuerprogramm ergibt sich die Notwendigkeit der Synchronisation mit bestimmten äußeren Bedingungen. Das Programm muß auf Betätigung einer Taste unverzüglich reagieren. Zudem sollen bestimmte Aufgaben wie das Auffrischen der Quellenparameter und der Bildschirmanzeige periodisch ausgeführt werden. Diese Aufgaben können nicht einfach durch

kontinuierliches Abfragen des Tastaturpuffers oder des Systemtimers gelöst werden, da dies unverhältnismäßig viel Rechenzeit in Anspruch nähme.

Aus Gründen der Betriebssicherheit soll das Programm außerdem auf Fehlermeldungen des Betriebssystems reagieren, die zum Abbruch des Programmes führen. So besteht die Möglichkeit, den aktuellen Quellenstatus noch vor dem Abbruch zu speichern, so daß ein Neustart des Programmes möglich ist, ohne daß der Betrieb der Quelle beeinträchtigt wird.

Das bereitgestellte Software-Interface zur Kommunikation des Steuerprogrammes mit anderen Programmen ist ebenfalls darauf angewiesen, daß auf übermittelte Kommandos unverzüglich reagiert wird.

Derartige Aufgaben werden vom UNIX-*Signal*-Mechanismus unterstützt, auf den nun eingegangen werden soll.

### 7.4.1 UNIX Signal-Mechanismus

Ein UNIX-*Signal* ist einem Hardware-Interrupt sehr ähnlich. Wenn ein C-Programm ein durch ein äußeres Event ausgelöstes *Signal* erhält, wird eine Betriebssystemroutine, der *Signal*-Handler, aufgerufen. Dieser kann das *Signal* ignorieren, das Programm beenden, oder in eine dem *Signal* zugeordnete Routine des Programmes verzweigen.

Einige *Signals* entsprechen Maschinen-Interrupts, z.B. die Floating-point-exception *SIGFPE*, die u.a. bei Division durch 0 auftritt; andere werden softwaremäßig durch das Betriebssystem (z.B. *SIGKILL*, Abbruch eines Prozesses) oder durch andere Programme erzeugt.

### 7.4.2 Programmierung mit Signals

Fast alle *Signals* - mit 2 Ausnahmen - können durch den Benutzer in eigene Routinen gelenkt werden. Die Ausnahmen sind *SIGKILL*, welches den Prozess gewaltsam abbricht, und *SIGSTOP*, welches den Prozess zeitweilig anhält.

Zur Manipulation von *Signals* stehen zwei C-Funktionen zur Verfügung: *signal* und *sigaction*. *signal* ist eine Kurzform von *sigaction*, bei der als Parameter nur die auszuführende Routine und das *Signal* angegeben werden müssen und die fehlenden Parameter mit einem Default-Wert besetzt werden. Dieser Default-Wert ist sehr stark betriebssystemabhängig und stellt auch einen wesentlichen Unterschied bei der Umsetzung des Programmes von LynxOS auf LINUX dar: Während die *Signal*-Funktion bei LynxOS die *Signal*-Routine statisch setzte, setzt LINUX diese als "One-Shot", d.h. nach einmaligem Ausführen der Routine setzt der *Signal*-Handler wieder seine Standard-Betriebssystemroutine zur Behandlung des *Signals* ein. Deshalb mußte unter LINUX die aufwendigere *sigaction*-Prozedur verwendet werden. Den Funktionen kann alternativ zur zugeordneten Routine die Konstante *SIG_IGN*, die zum Ignorieren des *Signals* führt, oder *SIG_DFL*, die die Standardroutine des Betriebssystems für das entsprechende *Signal* aufruft, übergeben werden.

Die *sigaction*-Funktion erhält als Parameter einen Action-Table. Dabei handelt es sich um eine Datenstruktur, in der die auszuführende Routine sowie verschiedene Flags angegeben werden, die ihr Verhalten beeinflussen.

#### Timer-Signals

Für das Steuerprogramm ist es notwendig, bestimmte Aktionen periodisch auszuführen. Hierzu bietet sich die Benutzung des internen Timers an.

UNIX stellt jedem Prozeß drei verschiedene Intervalltimer zur Verfügung. Diese Timer sind Zähler, die kontinuierlich dekrementiert werden. Ist ein Timer abgelaufen, sendet er ein *Signal* zum jeweiligen Prozeß.

Der erste Timer mißt die Realzeit , d.h. die tatsächlich vergangene Zeit. Er liefert das *Signal SIGALRM*. Ein zweiter Timer mißt die virtuelle Zeit, d.h. der Timer wird dekrementiert in der Zeit, in der der Prozeß ausgeführt wird. Dieser liefert das *Signal SIGVTALRM*. Der letzte Timer mißt die Summe aus virtueller und Realzeit und liefert *SIGPROF*.

Ein Timer wird durch die C-Funktion *setitime* gesetzt. Als Parameter wird außer dem zu benutzenden Timer eine Datenstruktur übergeben, die zwei Zeiten in Sekunden und Mikrosekunden enthält. Die erste Zeit wird direkt in den Timer geladen, die zweite gibt an, mit welchem Wert der Timer nach seinem Ablauf geladen werden soll. Ist der erste Wert gleich 0, stoppt der Timer, ist der zweite Wert gleich 0, läuft der Timer nur 1 mal und hält danach an.

Die theoretische Auflösung des Timers beträgt zwar 1 Mikrosekunde, die tatsächliche Auflösung wird jedoch durch das Betriebssystem bestimmt und beträgt bei LINUX und LynxOS 10 Millisekunden.

### Keyboard-Signals

Um eine komfortable Bedienung über die Tastatur zu gewährleisten, ist es erforderlich, daß das Programm unmittelbar auf einen Tastendruck reagiert, ohne daß lange Kommandozeilen eingegeben werden müssen.

Die Reaktion auf ein Tastatur-Event kann mit dem *Signal SIGIO* erfolgen. Dies muß jedoch erst mit der C-Funktion *fcntl*, die der Manipulation von File-Deskriptoren dient, freigegeben werden. File-Deskriptoren sind Datenstrukturen, deren Parameter das Ein- / Ausgabeverhalten einer Datei bestimmen.

Die Funktion *fcntl* erhält als ersten Parameter den File-Deskriptor, als zweiten einen Befehl und optional als dritten den entsprechenden Übergabewert. Auf das Terminal wird mit File-Deskriptor 0 zugegriffen. Zum Lesen und Schreiben der Deskriptorflags dienen die Befehle *F_GETFL* und *F_SETFL*. Zur Freigabe von SIGIO wird das Flag FASYNC gesetzt. Ebenfalls wird das Nonblocking-Statusflag *O_NDELAY* des File-Deskriptors gesetzt. Dieses verhindert, daß ein Lesen des Tastaturbuffers zum Blockieren des ausführenden Prozesses führt, falls gerade keine Daten anliegen [St92] .

Anschließend muß noch der Prozeß, der *SIGIO* empfangen soll, als Eigentümer des File-Sockets des Terminals bestimmt werden, damit das *Signal SIGIO* an den richtigen Prozeß gesendet wird. Das geschieht ebenfalls mit *fcntl*, wobei als Befehl *F_SETOWN* und als Wert die PID-Nummer des Prozesses übergeben werden.

### User-*Signals*

Das Steuerprogramm der Quelle verfügt über ein Software-Interface, durch das andere Programme dem Steuerprogramm Befehle senden können. Hierfür ist jedoch eine Synchronisation zwischen den beiden Programmen nötig. Das Client-Programm muß dem Steuerprogramm anzeigen, daß ein neuer Befehl vorliegt, und das Steuerprogramm muß die Ausführung dieses Befehles zurückmelden.

Zur Synchronisation zweier Prozesse untereinander stehen zwei User-definierte *Signals* zur Verfügung: *SIGUSR1* und *SIGUSR2*. Sie können mit der C-Funktion *kill* an einen beliebigen Prozeß geschickt werden, wobei die Prozeßidentifikationsnummer (PID) und das zu sendende *Signal* als Parameter übergeben werden.

**Abbruch-*Signals***

Eine Reihe von *Signals* führen normalerweise zum Abbruch des Programmes. Dies sind einerseits *Signals* , die speziell zu diesem Zwecke vom Benutzer an das Programm geschickt werden, wie z.B. *SIGINT*, aber auch Fehlermeldungen, wie *SIGSEGV* für einen Fehler im Speicherzugriff.

**Die pause-Funktion**

Ein wesentliches Argument für die Benutzung von *Signals* ist die Reduzierung der Rechenzeit. Das Programm soll nur dann aktiv werden, wenn ein *Signal* auftritt, ansonsten aber keine Rechenzeit in Anspruch nehmen. Hierfür stehen die C-Funktionen *pause* und *sleep* zur Verfügung. *pause* hält das Programm so lange an, bis das nächste *Signal* empfangen wird, und fährt erst nach dessen Abarbeitung mit dem Programm fort. *sleep* entspricht einem *pause*-Befehl, der aber gleichzeitig den Systemtimer aktiviert, so daß nach einer bestimmten Zeit der *pause*-Zustand automatisch beendet wird.

## 7.5 Shared Memory

Zur Übermittlung eines Befehls von einem Client-Programm an das Steuerprogramm wird die Technik des 'Shared Memory' (SHM) benutzt, bei dem mehrere Programme Zugriffsmöglichkeit auf einen gemeinsamen Speicherbereich haben.

Bei diesem Verfahren fordert ein Programm mit Hilfe der C-Funktion *shmget* einen Shared-Memory-Bereich an, der im Speicher angelegt wird. Als Rückgabewert erhält es eine ID-Nummer zur Identifikation des angeforderten Segmentes. Mit Hilfe der C-Funktion *shmat*, der die ID-Nummer übergeben wird, kann die Adresse des entsprechenden Segmentes ermittelt und so auf den zugeordneten Speicherbereich zugegriffen werden. Die ID-Nummer wird in einem File abgelegt. Ein Programm, das mit dem ursprünglichen Programm kommunizieren will, kann nun die ID-Nummer aus dem File lesen und nach obigem Verfahren ebenfalls auf den SHM-Bereich zugreifen. Auf diese Weise kann ein Datenaustausch zwischen beiden Programmen erfolgen.

Bei Beendigung der Programme wird der Shared-Memory-Bereich mit der Funktion *shmdt* wieder freigegeben. Die ID-Nummer wird mit der Funktion *shmctl* abgemeldet. Näheres siehe [Eg94]

## 7.6 Terminalprogrammierung unter UNIX

Die Bedienung über ein Text-Terminal ist ein wichtiges Element des Steuerprogrammes. Die Behandlung der Tastatur erfolgt, wie in Abschnitt 7.4.2 beschrieben, über das *Signal SIGIO*. UNIX wartet jedoch gewöhnlich auf den Abschluß einer Tastatureingabe durch die Return-Taste, bevor es die Eingabe verarbeitet. Damit die Betätigung einer Taste direkt an das Programm weitergegeben wird, ist eine spezielle Programmierung des Terminal-Treibers nötig.

**Terminal-IO:**

Unter UNIX werden die Zeichen, die über die Terminaltastatur eingegeben werden, nicht direkt zum Prozeß gesandt, sondern zuerst durch ein Interface-Programm, den Terminal-Driver, verarbeitet.

Normalerweise liest der Terminal-Driver Zeichen von der Tastatur, gibt sie auf dem Bildschirm aus und speichert sie in einem Puffer. Wird die Return-Taste gedrückt, wird der gesamte Puffer dem Prozeß übermittelt. Es gibt außerdem Zeichen, die gesondert behandelt werden. Einige Zeichen, wie z.B. BACKSPACE, haben Einfluß auf den Tastaturpuffer, andere, wie z.B. CTRL-C, lösen bestimmte *Signals* aus, die zum Prozeß geschickt werden.

Der Terminal-Driver arbeitet in einem von drei verschiedenen Modi: *cooked*, *cbreak* und *raw*. Der *Cooked*-Mode ist der oben beschriebene Standardmodus. Da die Eingabe erst verarbeitet wird, wenn die Return-Taste gedrückt wurde, ist in diesem Modus eine direkte Reaktion des Programmes auf eine einzelne Taste nicht möglich.

Im *Raw*-Modus werden die eingegebenen Zeichen (auch Sonderzeichen) vom Terminal-Driver nicht verarbeitet, sondern direkt an den Prozeß geschickt. Eine Darstellung am Bildschirm erfolgt nicht. Ein Programm, das im *Raw*-Modus arbeitet, reagiert nicht auf die üblichen Funktionstasten und läßt sich daher im Fehlerfall nicht auf herkömmliche Weise abbrechen.

Der *Cbreak*-Modus liegt zwischen beiden Modi: Zeichen werden ebenfalls, ohne gepuffert zu werden, an den Prozeß weitergegeben, aber auf dem Bildschirm dargestellt. Auch die meisten Sonderzeichen behalten ihre Wirkung.

Der Terminal-Driver kann über die UNIX/C-Funktion *ioctl* manipuliert werden. Ein komfortableres Verfahren bietet die Curses-Library, die auf praktisch allen UNIX-Systemen vorhanden ist.

### Die Curses-Library

Die Benutzung der Curses-Library bietet gegenüber der direkten Terminal-Programmierung folgende Vorteile:

- Die Library ist zu einer großen Anzahl verschiedener Terminals kompatibel. Der Programmierer kann einen Satz von Standardfunktionen verwenden, ohne sich über die Anpassung an die verschiedenen Terminalnormen Gedanken machen zu müssen.

- Die Library stellt alle wichtigen Funktionen zur Ausgabe auf dem Bildschirm zur Verfügung, egal, ob das benötigte Terminal diese explizit unterstützt oder nicht. Curses übersetzt die vom Programm gegebenen Befehle in die Kontrollsequenzen des Terminals, die nötig sind, um den gewünschten Effekt zu erzielen.

- Es sind ebenfalls Funktionen zur Manipulierung des Terminal-Drivers vorhanden, z.B. zum Setzen des *cbreak*-Modus oder Abschalten des Tastatur-Echos auf dem Bildschirm.

- Die Funktionen sind so ausgelegt, daß sie die Übertragungszeiten über das Netz minimieren. Die Curses-Funktionen speichern die Änderungen des Bildschirminhaltes zunächst intern und senden sie erst auf den Befehl *refresh* hin zum Terminal. Durch geschickte Programmierung kann so eine erhebliche Reduzierung der zum Terminal übertragenen Datenmenge erfolgen, die den Betrieb eines Programmes auch über sehr langsame Datenleitungen möglich macht.

Zur Initialisierung der Curses-Funktionen muß zunächst die Funktion *initscr* aufgerufen werden. *cbreak* schaltet das Terminal in den *cbreak*-Modus. Mit *noecho* läßt sich die Ausgabe der eingegebenen Zeichen auf dem Bildschirm abschalten. Die Funktion *nonl* verhindert, daß bei jeder Terminalausgabe ein Zeilenvorschub an das Terminal geschrieben wird. Die Curses-Funktionen *echo,nl,nocbreak* und *endwin* stellen den normalen Terminal-Modus wieder her.

Zur Ausgabe auf den Bildschirm wird vor allem die Curses-Funktion *printw* verwendet, die dem Curses-Äquivalent der normalen C-Ausgabefunktion *printf* entspricht. Der Cursor wird mit *move* plaziert, der Bildschirm mit *clear* gelöscht. Eine Übertragung der Änderungen auf dem Display zum Terminal erfolgt mit *refresh*.

Eine Eingabe vom Bildschirm erfolgt mit *scanw*. Zum Auslesen des Tastaturpuffers im cbreak-Modus wird die Funktion *getch* verwendet.

Die Curses-Bibliothek stellt verschiedene Funktionen zur Verwendung mehrerer unabhängiger Textfenster zur Verfügung. Da die meisten Programme mit nur einem Textfenster auskommen, kann der Funktionsumfang der Library mit der Compiler-Option *-DMINICURSES* entsprechend reduziert werden. Der Vorteil ist ein geringerer Umfang und eine höhere Geschwindigkeit des Programmes.

# Kapitel 8

# Aufbau des Steuerprogrammes

Das Steuerprogramm *piq* bildet den zentralen Teil der Steuersoftware. Es übernimmt die Ansteuerung von Quellencrate und Kontrollpult, Austausch und Verwaltung der Quellendaten, Bildschirmausgabe, Bearbeitung von Tastaturkommandos und Kommandozeilenbefehlen, Bereitstellung des Softwareinterfaces, Übermittlung externer CAMAC-Kommandos, Speicherung und Protokollierung der Daten und verschiedene Hilfsroutinen. Das unter C entwickelte Programm ist vollständig über ein einfaches VT100-Terminal bedienbar, die grafische Benutzeroberfläche wird mit einem separaten Client-Programm erzeugt.

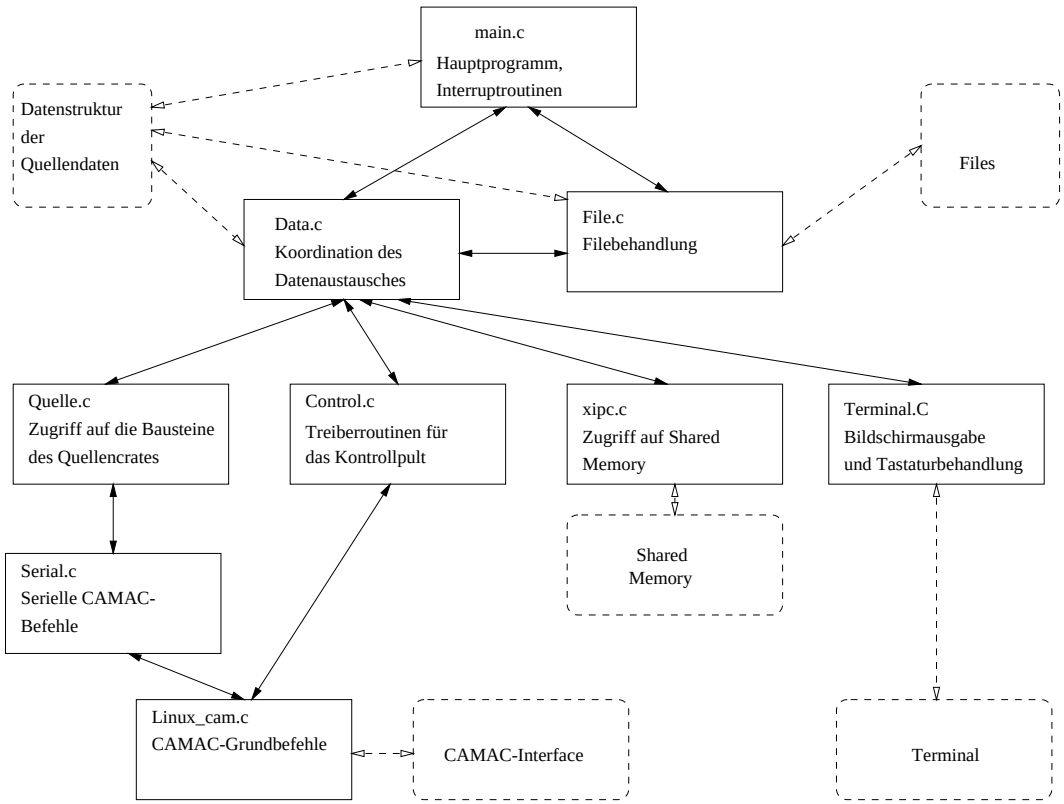


Abbildung 8.1: Hierarchie und Zugriffsrechte der einzelnen Quelltextmodule

Im folgenden sollen Aufbau und Funktionsweise des Programmes erläutert und der Quelltext des Programmes zur Erleichterung zukünftiger Erweiterungen dokumentiert werden.

Aus diesem Grunde wird stark auf die einzelnen im Programm enthaltenen Routinen und Variablen Bezug genommen.

Der Quelltext des Programmes wurde modular aufgebaut, d.h. die einzelnen Routinen wurden nach ihren Aufgaben getrennt in einzelnen Files untergebracht, die unterschiedliche Zugriffsrechte auf die Datenstrukturen haben (Abb. 8.1). Das hat entscheidende Vorteile bei einer zukünftigen Erweiterung des Programmes oder der Hardware. Das Modul *quelle.c* enthält z.B. Routinen zum Zugriff auf die Quellenhardware, die aber selbst nicht auf die Datenstruktur zur Speicherung der Quellendaten zugreifen, sondern ihre Werte als Parameter erhalten bzw. als Funktionsergebnis zurückliefern. Bei einer Änderung der Quellenbausteine muß man sich keine Gedanken über die innere Struktur des Programmes machen, sondern es reicht, einfache Routinen wie z.B. `read_channel(nr)` [1], der als Parameter die Kanalnummer übergeben wird und die den Wert des entsprechenden Kanals zurückgibt, zu implementieren.

Einige Hilfsroutinen der Steuersoftware gehen auf eine frühere Miniforschungsarbeit von U. Müssel [Mü94] zurück.

## 8.1 Programmablauf

Das Steuerprogramm hat nicht nur wie bei der alten Steuerung die Aufgabe des Datenaustausches zwischen Quelle und Kontrollpult, sondern wegen der verschiedenen Möglichkeiten der Dateneingabe (Terminal, Kontrollpult und externe Programme) auch die der Koordination, Umwandlung und Anzeige der Daten. Hinzu kommt, daß das Programm für die einzelnen Hardwarekomponenten ebenfalls die Aufgabe eines Software-Treibers übernehmen muß (vor allem für Pult und CAMAC-Karte).

Für die Koordination der verschiedenen Aufgaben des Programmes spielt dessen zeitlicher Ablauf eine entscheidende Rolle, da er darüber entscheidet, welchen Daten und Ereignissen Priorität bei der Bearbeitung eingeräumt wird. Das Hauptprogramm besteht aus zwei verschiedenen Schleifen. Je nach Betriebsmodus der Steuerung werden entweder kontinuierlich Daten zwischen Kontrollpult und Quelle übermittelt, oder das Programm verzweigt in einen Wartezustand, um auf die Betätigung einer Taste zu warten.

Die Betätigung einer Taste und die Übermittlung eines Steuerkommandos von einem anderen Programm werden in gesonderten Routinen behandelt, die durch die entsprechenden *Signals* aktiviert werden. Auf diese Weise erfolgt unabhängig von der Ausführung des Hauptprogrammes eine sofortige Reaktion auf das Ereignis.

Periodische Aufgaben wie die Aktualisierung der Bildschirmausgabe, das regelmäßige Speichern des aktuellen Quellenstatus in einer Datei und die zur Sicherheit erfolgende Auffrischung der Werte in den CAMAC-Ausgaberegistern der Quelle werden durch eine vom Systemtimer ausgelöste Routine ebenfalls unabhängig vom Hauptprogramm ausgeführt.

Einige untergeordnete Routinen dienen der Eingabe und Ausführung von Kommandozeilen und der Ausführung von externen CAMAC-Befehlen. Diese werden außerhalb des normalen Übertragungszyklus ausgeführt, um die Datenübertragung zur Quelle nicht zu beeinträchtigen.

### 8.1.1 Initialisierung

Die Initialisierungsroutine wird unmittelbar nach Start des Programmes aufgerufen. Sie prüft zunächst anhand der Existenz des Files *share.tmp*, in dem die Shared-Memory ID-

[1] Namen von Prozeduren und Variablen der Steuersoftware sind im folgenden in `Typewriter`-Schrift gesetzt.

Nummern gespeichert werden, ob bereits ein Steuerprogramm gestartet wurde. Dies dient dazu, gegebenenfalls den gleichzeitigen Zugriff zweier Steuerprogramme auf die Quelle zu verhindern.

Nun erfolgt die Auswertung der bei Start des Programmes aus der UNIX-Shell übergebenen Kommandozeilenoption, die das File bestimmt, aus dem die Initialisierungswerte für die Quellenparameter entnommen werden. Die Bedeutung der einzelnen Optionen ist in Anhang C.1 aufgeführt.

Die Initialisierung der Hardwarekomponenten und der Datenstrukturen übernimmt die Routine `init`. Sie ruft zunächst die Routine `init_values` auf, welche die Datenstruktur für die Quellendaten mit Default-Werten initialisiert. Dies stellt sicher, daß auch die Parameter, die nicht explizit im Setup-File gesetzt wurden, sinnvolle Startwerte erhalten. Nun erfolgt das Einlesen des Setup-Files und die Initialisierung der *Signal*-Routinen. Mit der C-Funktion *setuid 0* werden dem Programm Superuserrechte gewährt. Dies ist nötig, damit direkt auf die Ein-Ausgabeports, über die die CAMAC-Interface-Karte angesprochen wird, zugegriffen werden kann. Zuletzt werden die Routinen zur Initialisierung der CAMAC-Komponenten und des Terminals aufgerufen.

### 8.1.2 Hauptprogramm

Die alternativen Bedienungsmöglichkeiten des Programmes über Terminal oder Kontrollpult machen die Einführung verschiedener Betriebsmodi notwendig. Zum einen muß gewährleistet werden, daß während der Bedienung des Programmes über Terminal bzw. Pult die jeweils andere Bedienungsart gesperrt ist. Zum anderen verlangen die unterschiedlichen Hardwareeigenschaften eine unterschiedliche Behandlung durch die Software: Während bei der Bedienung durch das Kontrollpult die Stellung der Bedienungselemente kontinuierlich abgefragt werden muß, um eine möglichst hohe Übertragungsrate und damit eine unmittelbare Reaktion des Strahls auf Betätigung eines oder mehrerer Potentiometers zu erreichen, kann bei Bedienung über die Tastatur oder eine grafische Oberfläche von der UNIX-*Signal*-Technik Gebrauch gemacht werden, was zu einer erheblichen Verringerung der Rechenzeit führt.

Desweiteren ist ein Betriebsmodus erforderlich, in dem jede Änderung der Quellenparameter blockiert wird, um eine versehentliche Fehlbedienung auszuschließen.

Aus diesem Grunde benutzt das Programm drei verschiedene Betriebsmodi (Pult-, Terminal- und Lock-Modus), die in einer internen Variablen festgelegt werden. Nach erfolgreicher Initialisierung springt das Programm je nach Betriebsmodus in eine von zwei Schleifen, die so lange ausgeführt werden, bis eine Änderung des Betriebsmodus erfolgt. Im Pult-Modus erfolgt ein kompletter Übertragungszyklus, der die Werte der Bedienelemente des Pultes liest, in Sollwerte für die Quelle umrechnet und diese zur Quelle überträgt. Die 16 von der Quelle gelesenen Istwerte werden gespeichert und zu je einem Anzeigeinstrument des Pultes übermittelt. Im Terminal- und Lockmodus wird die Systemfunktion *pause* aufgerufen, die das Programm so lange anhält, bis ein *Signal* , beispielsweise durch Betätigung der Tastatur oder Ablauf des Systemtimers ausgelöst, empfangen wird.

Nach dem Schleifendurchlauf erfolgt die Bearbeitung des CAMAC-Stacks, der dazu dient, extern eingegebene CAMAC-Befehle außerhalb des Übertragungszyklus der Steuerung auszuführen, um den Steuerungsablauf nicht zu beeinträchtigen. Auf die Funktion dieses Stacks wird in Abschnitt 8.4.1 eingegangen.

Anschließend wird das Flag `CommandLineRequest` geprüft, das anzeigt, ob in der Zwischenzeit der Wunsch nach Eingabe einer Kommandozeile bestand, und gegebenenfalls in die Kommandozeilen-Eingaberoutine gesprungen.

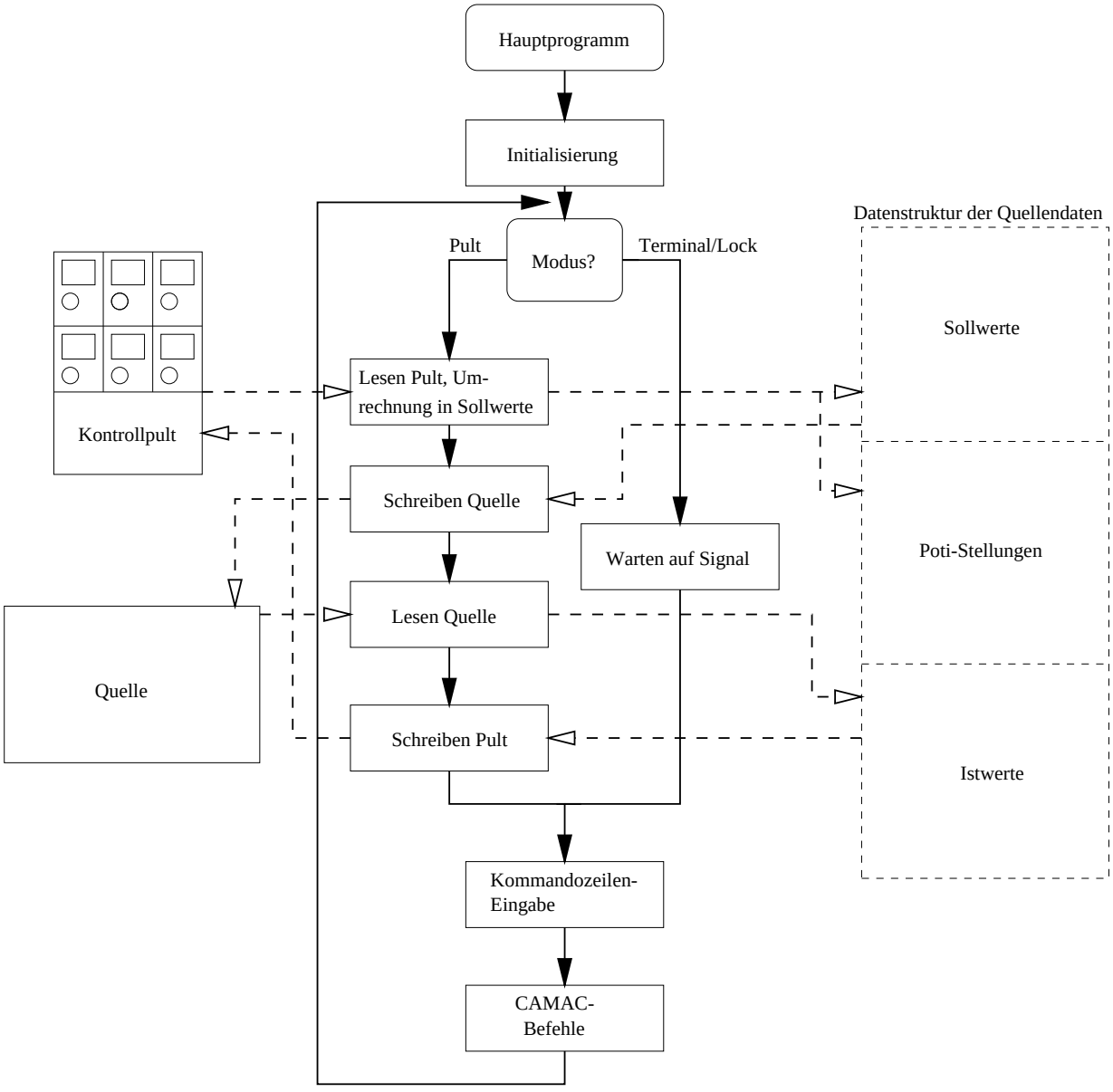


Abbildung 8.2: Ablauf des Hauptprogrammes (schematisch)

Das Flag wird jedesmal dann gesetzt, wenn der Benutzer durch Drücken der Taste 'i' des Keyboards die Eingabe einer Kommandozeile anfordert. Die *Signal*-Routine zur Bearbeitung der Tastatureingabe kann die Eingabe nicht direkt durchführen, da dies das Zurücksetzen des Terminals in den *cooked*-Modus und Abstellen des *SIGIO-Signals* erfordert. Wenn der Aufruf der *Signal*-Routine aus einer Routine erfolgt, die ihrerseits gerade auf den Terminal-Driver zugreift, können unvorhersehbare Effekte auftreten. Aus diesem Grunde setzt die *Signal*-Routine lediglich das angesprochene Flag, die eigentliche Eingabe mit Umschaltung des Terminalmodus geschieht aus dem Hauptprogramm heraus.

Der eingegebene Befehl wird durch den Kommandozeileninterpreter bearbeitet, der auch zur Ausführung der über das Softwareinterface oder das Setup-File eingegebenen Befehle dient. Eine Liste der möglichen Befehle befindet sich in Anhang D.

Nach Abarbeitung dieser Vorgänge kehrt das Programm zur Abfrage des Betriebsmodus zurück und verzweigt erneut in eine der beiden Schleifen.

### 8.1.3 Signal-Routinen

Für die Behandlung der Tastatur ist eine Verwendung des UNIX *Signal*-Mechanismus zweckmäßig. Durch eine entsprechende *Signal*-Routine kann eine sofortige Reaktion auf Betätigung einer Taste erfolgen, ohne daß die Tastatur kontinuierlich abgefragt werden muß und damit eine hohe Rechenzeit in Anspruch genommen wird.

Von der gleichen Technik macht das Softwareinterface Gebrauch, das der Ausführung von Steuerbefehlen, die von anderen Programmen übermittelt wurden, dient.

Es existiert außerdem eine durch das *SIGALRM-Signal* des internen Timers ausgelöste Routine zur periodischen Aktualisierung des Quellen- und Pultstatus und der Bildschirmanzeige.

#### Timer

`time_dep` ist die Routine, die bei einem Timer-*Signal* aufgerufen wird. Sie setzt zunächst das Flag `TimerIRQ`. Sollte während der Ausführung der Routine eine weitere *Signal*-Routine aufgerufen werden, kann sie anhand dieses Flags die Ausführung der Timer-*Signal*-Routine feststellen und Zugriffe, die diese stören könnten, unterlassen.

Anschließend wird die Routine `dump_perhaps` aufgerufen, die prüft, ob eine Protokollierung der Quellenwerte erfolgen soll. Dazu wird ein interner Zähler hochgezählt und bei Überschreiten eines festgelegten Maximalwertes die Protokollierung ausgeführt.

Ein weiteres Flag, welches verhindert, daß sich verschiedene *Signal*-Routinen gegenseitig bei der CAMAC-Übertragung stören, ist das Flag `Block_CAMAC`. Falls `Block_CAMAC` nicht gesetzt ist, eine CAMAC-Übertragung also nicht durch andere Routinen blockiert wird, setzt die Timer-Routine ihrerseits das Flag und führt eine Übertragung aller Werte von und zur Quelle und ggf. zum Kontrollpult durch. Anschließend wird `Block_CAMAC` wieder freigegeben. Da alle Quellenparameter in einer internen Datenstruktur gespeichert werden, führt das Ausbleiben einer CAMAC-Übertragung bei gesetztem `Block_CAMAC`-Flag nicht zum Verlust irgendwelcher Informationen. Geänderte Werte werden beim nächsten CAMAC-Übertragungszyklus zur Quelle geschrieben. Im Pult-Modus beträgt die Frequenz der CAMAC-Zyklen ca. 35 Hz, im Terminal-Modus ca. 2 Hz. Im Terminal-Modus ist jedoch die Zeit zwischen 2 CAMAC-Zyklen im Verhältnis zur Dauer eines Zyklus so groß, daß eine Überschneidung der CAMAC-Zugriffsversuche durch verschiedene *Signal*-Routinen in der Praxis keine Rolle spielt.

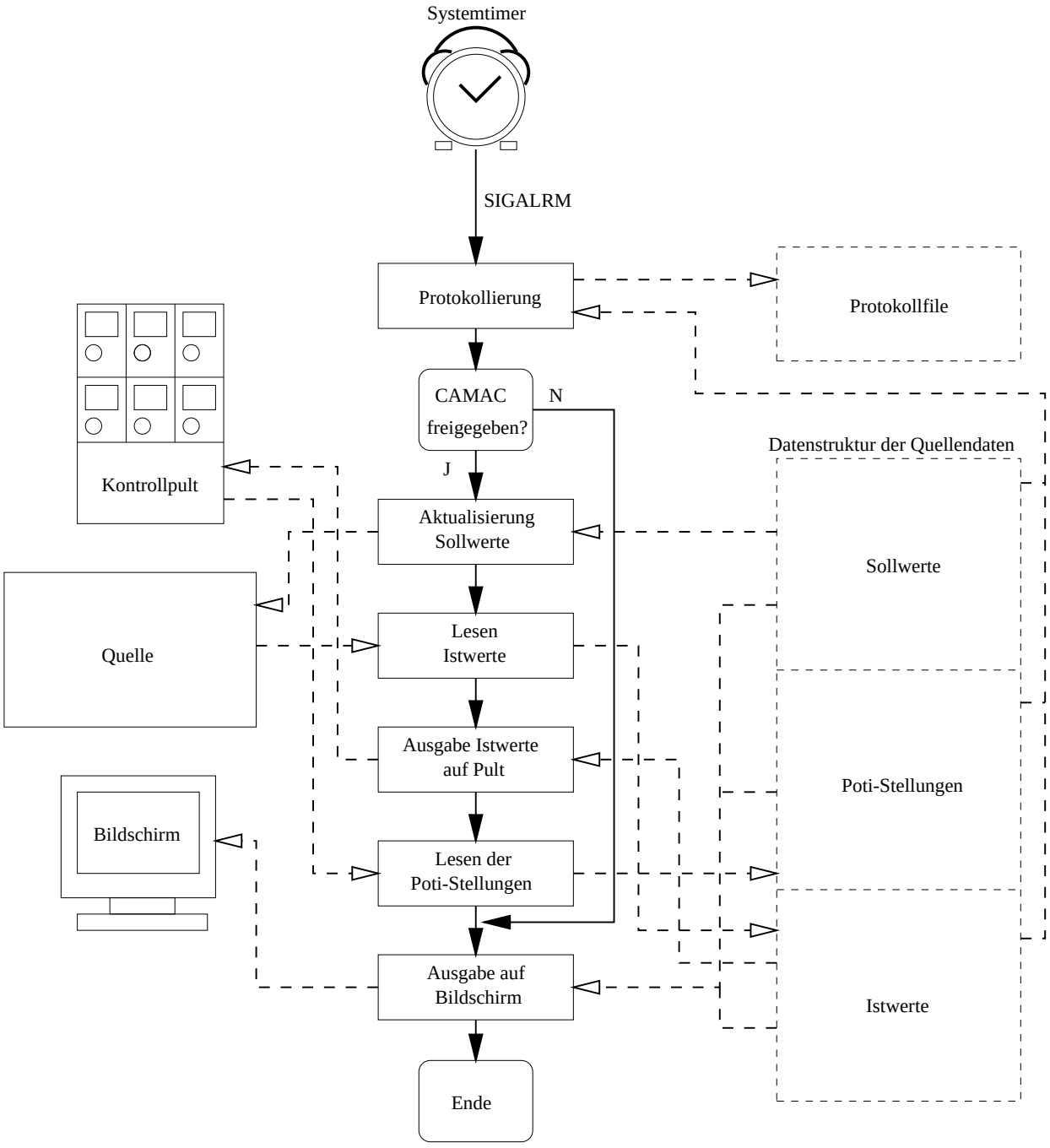


Abbildung 8.3: Timer-Routine

Mit Hilfe des Flags `KeyIRQ` wird geprüft, ob beim Sprung in die Timerroutine gerade ein Keyboard-*Signal* ausgeführt wurde. Falls nicht, gibt die Routine ihrerseits die geänderten Werte auf dem Terminal aus. Nun werden Key- und Timer-*Signal* neu gestartet, das Flag `TimerIRQ` zurückgesetzt und das File *piq_actual* geschrieben, welches den aktuellen Status der Quelle enthält und das vom Überwachungsprogramm *piqwatch* benutzt wird. Schließlich erfolgt das Kopieren sämtlicher Quellendaten in einen Shared-Memory-Bereich, um Client-Programmen den Zugriff zu ermöglichen.

### Keyboard

Der Aufruf der Routine zur Bearbeitung des Keyboards geschieht durch das *Signal SIGIO*. Das Programm kann zwischen zwei verschiedenen Tastaturbelegungen unterscheiden, zwischen denen vom Benutzer manuell umgeschaltet werden kann. Bei der einen Belegung sind jedem Kanal zwei Tasten zur Veränderung der Werte der analogen Kanäle zugeordnet, bei der anderen entsprechen den Tasten verschiedene Funktionen.

Die im Programm definierten Routinen `enable/disable key_irq` dienen zum An- bzw. Abschalten des Keyboard-*Signals* . Die Routinen setzen die *Signal*-Routine für *SIGIO* auf die Routine `keypress` bzw. auf *SIG_IGN*.

Die Routine `keypress` holt den Wert der betätigten Taste aus dem Tastaturpuffer und liest diesen anschließend leer, um evt. nachfolgende Sonderzeichen zu eliminieren. Danach verzweigt sie je nach gewählter Tastaturbelegung in eine von zwei Routinen, die den Tasten verschiedene Funktionen zuordnen und diese ausführen.

Einige dieser Routinen haben lediglich Einfluß auf interne Parameter des Programmes, wie z.B. den Betriebsmodus. Bei den Routinen, die eine Werteänderung zur Folge haben, geschieht dies über die Routine `change_value`, die in Abschnitt 8.2.4 beschrieben ist und die die Übermittlung der Werteänderung zur Quelle und die Ausgabe auf dem Bildschirm regelt.

### Client-Prozesse

Das *Signal SIGUSR2* wird von externen Prozessen an das Steuerprogramm geschickt, um anzuzeigen, daß ein neuer Befehl im Shared-Memory-Bereich des Softwareinterfaces vorliegt. Bei Erhalt des *Signals* verzweigt das Steuerprogramm in die Routine `x_irq`. Diese liest zunächst aus dem Shared-Memory-Bereich die Prozeßidentifikationsnummer (PID) des aufrufenden Prozesses, die am Anfang des Befehlsstrings als ASCII-String abgelegt ist. Der Rest des Befehlsstrings wird dem Command-Line Interpreter übergeben, der den Befehl ausführt und sein Ergebnis in einen zweiten SHM-Bereich schreibt. Nach erfolgter Abarbeitung des Befehls wird mit Hilfe der ermittelten PID an den Client-Prozess das *Signal SIGUSR1* gesendet, um ihm das Vorliegen des Befehlsergebnisses zu signalisieren. Die Routine `out_changes` wird aufgerufen, die evt. aufgetretenen Änderungen der Quellenwerte auf dem Bildschirm anzeigt.

### Programmabbruch

Zum Verlassen des Programmes dient die Prozedur `goodbye`, die durch ein Terminate-*Signal* (Ctrl-C) oder den Kommandozeilenbefehl 'quit' aufgerufen wird. Alle gängigen Fehler-*Signals* , die zum Abbruch des Programmes führen würden, wie z.B. *Floating-Point Exception* oder *Segmentation Fault*, führen ebenfalls zum Aufruf dieser Routine. Zunächst wird eine Speicherung des aktuellen Quellenstatus in das File *piq_panic* durchgeführt. Ein

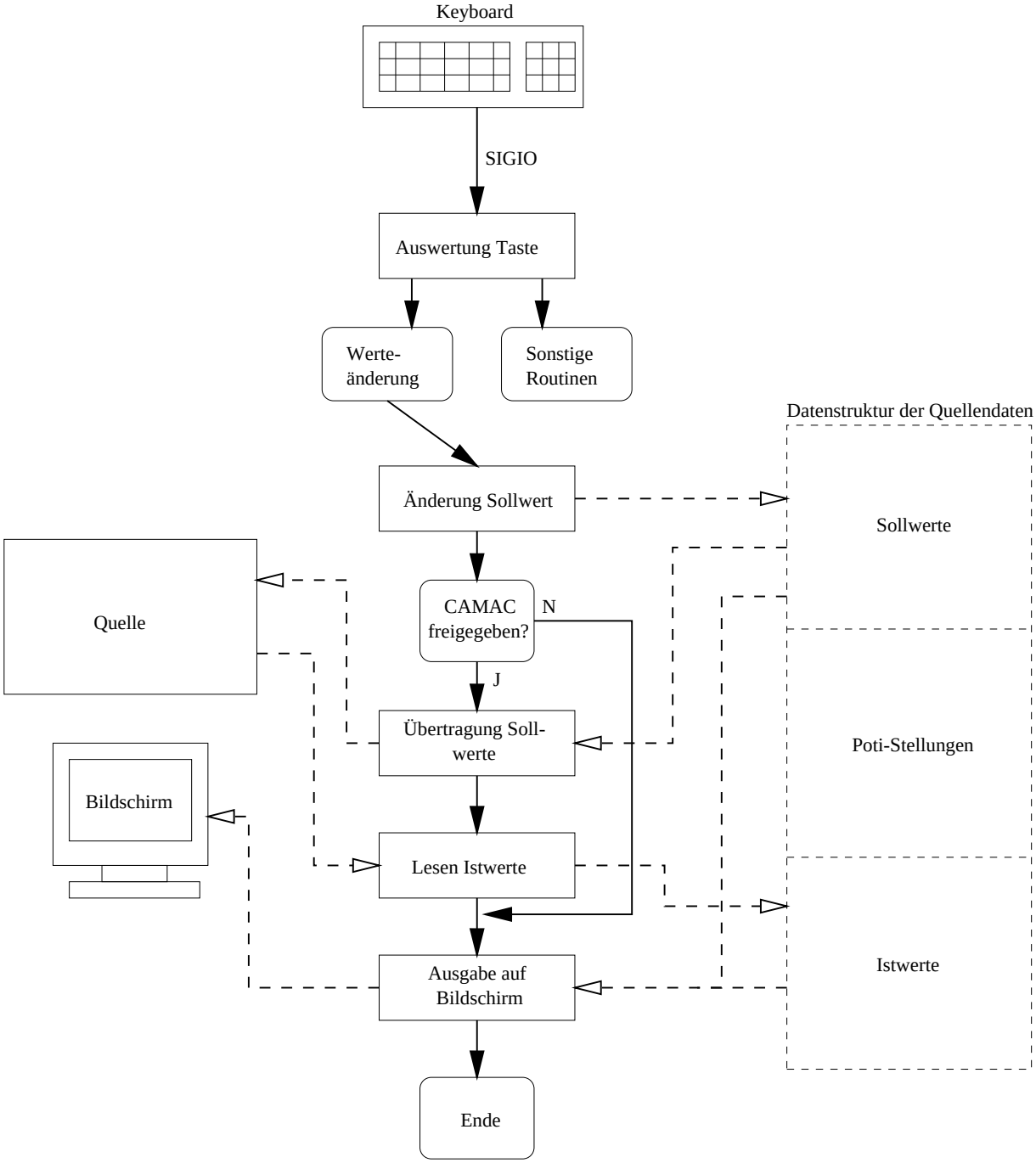


Abbildung 8.4: Keyboard-Routine einschließlich Aufruf der Routine zur Werteänderung ‘change_value’.

Neustart des Programmes ist auf diese Weise möglich, ohne daß die Quellenparameter beeinflußt werden. Die abgefangenen *Signals* sind: *SIGABRT, SIGHUP, SIGIOT, SIGQUIT, SIGSEGV, SIGTERM, SIGTRAP, SIGINT*. Anschließend wird der Timer-IRQ abgeschaltet und die einzelnen Routinen zur Freigabe der entsprechenden Speicherbereiche aufgerufen. Danach wird das Programm beendet.

## 8.2 Datenübertragung

Das Steuerprogramm erhält laufend Daten von der Quelle und den angeschlossenen Geräten (Pult, Terminal, Benutzeroberfläche), die gespeichert, umgerechnet und an die anderen Geräte gesendet werden müssen. Diese Daten müssen zum einen zwischen PC und den entsprechenden Geräten ausgetauscht werden, wozu Treiberroutinen im Steuerprogramm integriert sind. Desweiteren müssen die Reaktionen auf Änderung der Daten durchgeführt und koordiniert werden, damit verschiedene Eingabemedien sich nicht gegenseitig stören können. Zudem müssen auftretende Übertragungsfehler erkannt und eliminiert werden.

### 8.2.1 Kommunikation mit der Quelle

Das Programm enthält verschiedene elementare Routinen, die zur Datenübermittlung zwischen Quelle und Rechner und zur Initialisierung der CAMAC-Bausteine des seriellen Highways dienen. Sie sind in der Quelltextdatei *quelle.c* implementiert. Diese Routinen dienen nur dem elementaren Zugriff auf die Quellenhardware, ein Datenaustausch mit anderen Geräten und ein Speichern der Werte in den Datenstrukturen des Programmes findet in anderen Routinen statt.

Die Routinen greifen auf die CAMAC-Bausteine der Quelle über die in *linux_cam.c* definierten C-Routinen zu, die ihrerseits über die IO-Ports des Rechners auf das CAMAC-Interface zugreifen. Bei Zugriff auf das Quellencrate werden auch die Routinen aus *serial.h* benutzt, die die als Parameter übergebenen CAMAC-Kommandos in die für die serielle Übertragung geeignete Form umwandeln und sie ebenfalls mit Hilfe der CAMAC-Grundbefehle an den Serial Highway-Driver übermitteln.

Den einfachen Routinen aus *quelle.c* übergeordnet sind die Routinen in der Datei *data.c*, die den Datenaustausch und die Synchronisation zwischen den einzelnen Hardwarekomponenten sowie die Speicherung und Verarbeitung der Daten in den zentralen Datenstrukturen des Programmes übernehmen.

#### Initialisierung des Quellencrates

Die Routine `init_piq` führt eine Initialisierung der CAMAC-Bausteine der Quelle durch. Zunächst wird aus einem File die aktuelle CAMAC-Konfiguration eingelesen, d.h. Branch und Crate-Nr. des Kontrollcrates, Station des SHD und des IO440-Registers sowie Crate-Nummer des seriellen Crates. Anschließend wird mit `init_piq_camac` das CAMAC-System, d.h. die PC-CAMAC-Karte, der serielle Highway-Driver und der serielle Crate-Controller, wie in Abschnitt 4.3.2 beschrieben, initialisiert. Nun werden mit der Routine `init_piq_adresses` die Datenstrukturen für die CAMAC-Module auf der Quelle angelegt. Deren CAMAC-Adressen sind in der Header-Datei *config.h* festgelegt.

Mit der Routine `init_piq_modules` werden schließlich die Konfigurationsbefehle für den Multiplexer und den ADC an die Quelle gesendet. Der ADC wird durch Schreiben des Wertes 0x2D6 ins Statusregister initialisiert. Dies konfiguriert den ADC für einen Spannungsbereich

von $\pm 5V$ und schaltet die LAM-Erzeugung und den Rauschunterdrückungsfilter aus. Der Conversion Delay, der eine Störung des ADC durch Umschalten des Multiplexers verhindert, und der Scan+1-Mode, der nach erfolgter Konversion ein Signal zum Multiplexer schickt, welches diesen zur Kanalweiterschaltung veranlaßt, werden eingeschaltet. Diese Konfiguration entspricht der des alten Systems. Der Multiplexer wird in den Recycle-Mode geschaltet, um auf das Signal des ADC hin den Kanal zyklisch weiterzuschalten.

### Digitale Daten

Die Strahlpolarisation `pol`, die Umlenkung des Strahls auf den Quellencup oder zum Tandem `dir` und der 'Quench all'-Betriebszustand `quench` werden über das duale Eingaberegister JRE und das duale Ausgaberegister JRS im Quellencrate gesteuert.

Den digitalen Sollwerten entsprechen aus historischen Gründen [An76] jeweils zwei Bits im Register JRS0. Die Umrechnung der Sollwerte in das entsprechende Bitmuster und das Schreiben in das Register JRS0 des Quellencrates geschieht in der Routine `q_write_digital`, der als Parameter die Werte für `pol`, `dir` und `quench` übergeben werden. Zur besseren Lesbarkeit des Quelltextes sind im File *global.h* Konstanten für die möglichen Werte für `pol`, `dir` und `quench` definiert.

Die Istwerte entsprechen ebenfalls bestimmten Bits im Eingaberegister JRE0. Die Istwerte werden mit der Routine `q_read_all_digital` ermittelt, die den Wert des Registers JRE0 einliest und die entsprechenden Werte für `pol` und `dir` aus dem Bitmuster ermittelt. Zurückgegeben werden die Werte von `pol` und `dir` und der komplette Inhalt des Eingaberegisters JRE0.

Die Routine `q_get_extint` liefert als Ergebnis den Inhalt des Registers JRE1 zurück, welches den extern/intern-Status der Netzgeräte enthält.

### Analoge Daten

**Sollwerte:** Den analogen Sollwerten entsprechen einem der beiden Teilregister der dualen Ausgaberegister DOR 1-8. Die dort gespeicherten Werte werden auf DAC-Platinen in Steuerspannungen für die Netzgeräte der Quelle umgewandelt.

Das Schreiben eines digitalen Sollwertes geschieht mit der Routine `q_write_channel`, der die Kanalnummer, der neue Sollwert und das Flag `ext_dmd` übergeben werden. Die Routine spiegelt zunächst die 12 Bits des Sollwertes. Dies ist wegen der internen Verdrahtung auf der Quelle nötig, da die ursprüngliche Steuerung die Sollwerte ebenfalls in gespiegelter Form übermittelte. Ist das Flag `ext_dmd` gleich 1, wird zusätzlich Bit 16 im Sollwert gesetzt. Dies bewirkt auf der Quelle eine Anforderung des externen Bedienungsstatus des entsprechenden Netzgerätes. Nun wird aus der Kanalnummer die Adresse und Subadresse des entsprechenden Registers ermittelt und der Sollwert übertragen.

**Istwerte:** Die von den Netzgeräten erzeugten analogen Rückmeldespannungen werden über einen CAMAC-Multiplexer-Baustein auf einen CAMAC-ADC gelegt und dort digitalisiert.

Das Auslesen des digitalisierten Wertes geschieht mit den Routinen `q_read_channel`, die den Wert des als Funktionsparameter übergebenenen Kanals liest, und `q_read_inc_channel`, die den Wert des aktuellen Kanals des Multiplexers liest. Multiplexer und ADC befinden sich im Scan-Through-Mode, bei dem nach jedem Auslesen des ADC der Kanal automatisch inkrementiert und eine neue Konversion gestartet wird. `q_read_channel` setzt daher zunächst die Kanalnummer-1 und liest den ADC aus, um die Konversion in Gang zu setzen.

Beide Routinen warten auf das Ende der Konversion und lesen dann das Ergebnis des ADC aus. Dieses wird mit Hilfe der Routine `adc2int` ausgewertet.

Die unteren 11 Bits des ADC-Wertes enthalten den digitalisierten Wert der Rückmeldespannung des Kanals. Bit 12 bedeutet ein negatives Vorzeichen, was jedoch physikalisch unsinnig ist und daher auf einen Fehler hindeutet. In diesem Fall wird von der Routine der Wert -1 zurückgegeben. Bit 13 zeigt einen Wertüberlauf an; in diesem Falle wird der Wert -2 zurückgegeben.

Die Routine `q_read_inc_channel` liest zusätzlich die aktuelle Kanalnummer des Multiplexers aus und gibt diese zurück.

### 8.2.2 Kommunikation mit dem Kontrollpult

Die Kommunikation mit dem Kontrollpult erfolgt über das $2 \times 24$-Bit CAMAC-Ein-/Ausgaberegister IO440 der Firma Hytec, das sich im Steuercrate befindet. Die unteren 16 Bits des Ein- und Ausgaberegisters entsprechen den beiden Datenbussen des Kontrollpultes. Die anderen Bits des Ausgaberegisters entsprechen den Strobe- und Enable-Signalen der Treiberplatinen. Die restlichen Bits des Eingaberegisters enthalten Statusbits und die Kanalnummer des Pult-ADCs.

Zum Lesen und Schreiben dienen die Routinen im Quelltextfile *control.c*.

`init_io` initialisiert das IO440-Register. Sollte das Register nicht existieren, wird das Flag `No_Io440` gesetzt. Dadurch ist ein Betrieb der Steuerung auch ohne das Kontrollpult möglich. Befehle, die auf das Register zugreifen, werden bei gesetztem Flag einfach ignoriert.

- `c_read_switches` liest die Stellung der Kippschalter für die digitalen Quellenparameter auf dem Kontrollpult ein. Hierzu ist nur die Ausgabe eines einzelnen Enable-Signals M1-1 nötig, das die Werte auf die W-Datenleitungen legt. Der gelesene Wert wird durch die Routine in den entsprechenden `pol`, `dir` und `quench`-Wert konvertiert.

- `c_set_leds` gibt die Werte der digitalen Quellenparameter auf den LEDs in der Zentrale des Kontrollpultes aus. Hierzu wird der Wert aus JRE0, der der Routine als Parameter übergeben wird, auf die R-Leitungen gelegt und das Strobe-Signal Z1 gesetzt.

- `c_set_ei_leds` steuert die Extern/Intern LEDs der Fernbedienungseinschübe. Der Wert aus JRE1, der der Routine als Parameter übergeben wird, wird auf die R-Leitungen gelegt und das Strobe-Signal Z5 gegeben.

- `c_write_ana_disps(nr, value)` gibt den Wert `value` auf dem Anzeigeinstrument `nr` aus. Hierzu wird zunächst die Nummer des Kanals in die unteren 4 Bits der Read-Leitung geschrieben und durch das Signal Z3 übernommen. Danach werden die höchsten 8 Bit in die Read-Leitung R4-R11 geschrieben und durch Strobe Z4 auf das Display übertragen.

- `c_read_poti(*nr, *value)` liest den Wert des aktuellen Potentiometers aus und speichert diesen in `value`, die Nummer des Potentiometers wird in `nr` gespeichert. Die Nummer des Potentiometers wird durch die aktuelle Kanalnummer des Multiplexer im ADC des Kontrollpultes bestimmt, der sequentiell weiterschaltet. Der Wert der Kanalnummern liegt an den Adressleitungen A-E des Kontrollpultes an.

### 8.2.3 Kommunikation mit dem Terminal

#### Keyboard

Auf die Behandlung des Keyboards wurde in Abschnitt 8.1.3 eingegangen.

#### Bildschirm

Die Ausgabe auf dem Bildschirm bedient sich der Curses-Library, die in der Routine `init_terminal` initialisiert wird. Diese Routine initialisiert auch das Keyboard durch Schalten in den cbreak-Modus.

Alle Bildschirmausgaben erfolgen über in der Datei *terminal.c* implementierte Routinen. Allein diese Routinen enthalten Informationen über die Ausgabeposition der Werte in der Bildschirmmaske, so daß eine Änderung des Bildschirmaufbaus mit relativ geringem Aufwand möglich ist.

Diesen einfachen Routinen sind die Routinen `out_all` und `out_changes` aus *data.c* übergeordnet. `out_all` gibt sämtliche Daten auf dem Bildschirm aus, `out_changes` gibt nur die Daten aus, bei denen das Flag `chflag` gesetzt ist. Dieses Flag existiert sowohl für jeden Analogkanal als auch für die digitalen Werte und wird bei jeder Änderung eines Wertes gesetzt. Alle Ausgaberoutinen setzen nach erfolgter Ausgabe das Flag für die entsprechenden Daten zurück.

### 8.2.4 Datenaustausch

| Feld | Typ | Bedeutung |
|---|---|---|
| name | String | Name des Kanals |
| r_unit | String | Einheit des Istwertes |
| w_unit | String | Einheit des Sollwertes |
| format | String | Formatstring zur Ausgabe |
| v_poti | Int | Absolutwert des Pult-Potentiometers |
| w_piq | Int | Sollwert |
| r_piq | Int | Istwert |
| pos | Int | Zeilenposition des Kanals in der Bildschirmmaske |
| scale_read_a | Double | Eichparameter des Istwertes nach $a + b * x$. |
| scale_read_b | Double | Eichparameter Istwert |
| scale_write_a | Double | Eichparameter Sollwert |
| scale_write_b | Double | Eichparameter Sollwert |
| chflag | Int | Signalisiert Werteänderung des Kanals |
| dumpflag | int | Signalisiert, daß der Kanal protokolliert werden soll |
| min | Int | Untere Schwelle für den Sollwert in Skt. |
| max | Int | Obere Schwelle für den Sollwert in Skt. |
| maxalarm | Int | Obere Alarmschwelle für den Istwert |
| minalarm | Int | Untere Alarmschwelle für den Istwert |
| res | Int | Auflösung der Potentiometer in Zweierpotenzen. |
| flags | Int | Flags für besondere Eigenschaften des Kanals |

Tabelle 8.1: Felder der Datenstruktur `piq_channel`

Die Koordination des Datenaustausches zwischen Steuerprogramm, Quelle, Terminal und Kontrollpult sowie die Speicherung der Daten erfolgt über Routinen in der Quelltextdatei *data.c* . Diese greifen auf zwei Datenstrukturen zu, in denen die Quellenwerte gespeichert werden, und rufen die oben besprochenen Routinen auf, die die Daten zu den Hardwarekomponenten übertragen. Die Routinen stellen ebenfalls die Koordination der verschiedenen Ein- und Ausgabemedien sicher; so werden nur die Werte in der Bildschirmausgabe aktualisiert, die sich seit der letzten Aktualisierung geändert haben. Auch wird gewährleistet, daß nur die durch den aktuellen Betriebsmodus zugelassene Werteeingaben eine tatsächliche Änderung des Sollwertes zur Folge haben.

### Datenstrukturen

**Digitale Werte** Die digitalen Sollwerte sind in der Datenstruktur `piq_status` in den Feldern `s_pol`, `s_dir`, `s_quench` gespeichert, die entsprechenden Ist-Werte in den Feldern `pol`, `dir`, `quench` derselben Datenstruktur. Zusätzlich ist der komplette Inhalt des CAMAC-Registers JRE0, das die digitalen Istwerte enthält, im Feld `jre0` enthalten. Der Inhalt des CAMAC-Registers JRE1, welches den Extern/Intern-Status der Netzgeräte enthält, ist im Feld `jre1` gespeichert.

Zusätzlich enthält die Datenstruktur das Feld `mode`, das den aktuellen Eingabemodus (Pult, Terminal, Lock) des Programmes enthält, das Flag `bit16` für die Anforderung des externen Status, und das Flag `chflag`, welches bei einer Änderung der digitalen Daten gesetzt wird.

**Analoge Werte** Die Werte der analogen Daten werden im Array `piq_channel` vom Typ `channel_struct` gespeichert. Das Format der Datenstruktur `channel_struct` ist in Tabelle 8.1 dargestellt.

### Änderung der digitalen Werte

Eine Änderung der digitalen Sollwerte kann durch einen Tastendruck im Keyboard-Modus, die Betätigung eines Schalters im Pult-Modus oder durch den Befehl eines anderen Programmes ausgelöst werden. Die Routinen, die diese Ereignisse behandeln, führen alle Änderungen der digitalen Werte über die Routine `set_digital` in *data.c* aus. Zur Änderung eines einzelnen digitalen Parameters existieren die Routinen `set_pol`, `set_dir` und `set_quench`, die ihrerseits `set_digital` aufrufen.

`set_digital` prüft zunächst, ob die neuen Sollwerte von den alten abweichen. Falls ja, speichert es die neuen Sollwerte in der Datenstruktur und setzt das Flag `chflag`, damit die neuen Werte bei nächster Gelegenheit ausgegeben werden. Nun wird geprüft, ob die CAMAC-Ausgabe durch eine andere *Signal*-Routine gesperrt ist. Falls nicht, wird der neue Sollwert sofort zur Quelle übertragen, der neue Quellenstatus gelesen und auf Bildschirm und Kontrollpult angezeigt. Falls doch, werden die neuen Werte zunächst nicht übertragen. Da die neuen Sollwerte jedoch in der Datenstruktur gespeichert wurden, erfolgt die Übertragung zur Quelle automatisch beim nächsten Timer-*Signal* .

Im Falle der digitalen Werte werden auf Bildschirm und Kontrollpult nur die Istwerte angezeigt. Eine separate Anzeige des Sollwertes ist nicht notwendig, da bei korrekter Funktion der Steuerung Ist- und Sollwert stets übereinstimmen. Eine Nichtübereinstimmung zwischen digitalem Ist- und Sollwert wird jedoch sofort angezeigt und durch ein akustisches Alarmsignal (Keyboard-Glocke) unterstützt.

Zur Umschaltung des Eingabemodus dient die Routine `set_mode`. Bei Umschaltung auf den Pult-Modus wird die aktuelle Stellung der Potentiometer im Kontrollpult eingelesen und auf die aktuellen Sollwerte kalibriert, so daß unabhängig von der Stellung der Potentiometer die aktuellen Sollwerte beibehalten werden. Ist das Flag `POTI_ABS` gesetzt, wird der aktuelle Sollwert gleich dem Absolutwert des Potentiometers gesetzt.

### Änderung der analogen Werte

`change_value` ist die zentrale Routine zur Änderung eines analogen Wertes. Übergeben werden die Kanalnummer, die Werteänderung, das Eingabemedium (Potentiometer oder Keyboard) sowie ein Flag, das anzeigt, ob der Änderungswert absolut oder relativ zu verstehen ist. Alle analogen Werteänderungen, die im Programm erfolgen, benutzen diese Routine. Ein Eingriff in die Umrechnung einzelner Werte oder eine globale Überwachung der Werte sollte immer innerhalb dieser Routine geschehen, um auf alle im Programm auftretenden Werteänderungen Einfluß zu haben.

Die Routine übernimmt auch die Aufgabe der Überprüfung des Eingabemodus: eine tatsächliche Werteänderung erfolgt nur, wenn das Eingabemedium, von welchem die Werteänderung angefordert wurde, durch den aktuellen Bedienungsmodus des Programmes zugelassen wird. Bei Betätigung der '+'-Taste im Pult-Modus wird zwar durch die Keyboard-Routine eine Erhöhung des aktuellen Wertes über die `change_value`-Routine angefordert, diese aber durch die Routine abgeblockt.

Die Routine prüft zunächst, ob der richtige Eingabemodus und eine gültige Kanalnummer vorliegen. Dann wird überprüft, ob der neue Wert eine der Maximal- bzw. Minimalgrenzen überschreitet. Das Werteänderungsflag für den entsprechenden Kanal wird gesetzt und der neue Sollwert entsprechend dem Relativ- oder Absolutmodus errechnet und gespeichert. Anschließend wird die Möglichkeit einer CAMAC-Übertragung geprüft und ggf. durchgeführt. Der aktuelle Istwert wird gelesen und auf Bildschirm und Pult angezeigt.

Zur Umrechnung der Soll- und Istwerte von Skalenteilen in physikalische Einheiten dienen die Routinen `ieich` und `seich` die sich einer linearen Eichfunktion bedienen. Die Parameter der Eichfunktion werden im File *piq_setup* festgelegt und befinden sich in der Datenstruktur `piq_channel`(s.o.). `uieich` und `useich` sind die Umkehrfunktionen zu `ieich`, `seich` und dienen zur Rückrechnung von dezimal eingegebenen Werten auf Sollwerte.

**Aktualisierung der Werte:** Insbesondere zum periodischen Auffrischen der Werte dienen die folgenden Routinen: `read_all_piq` und `refr_piq` dienen zur Auffrischung der Quellenwerte auf der Quelle bzw. in der Datenstruktur. `read_all_piq` liest mit Hilfe der in *quelle.c* definierten Routinen alle Werte von der Quelle ein und speichert sie in der Datenstruktur `piq_channel`. Bei einer Änderung des Wertes wird `chflag` gesetzt, so daß der Wert zum nächstmöglichen Zeitpunkt ausgegeben wird. `refr_piq` schreibt alle in der Datenstruktur gespeicherten Sollwerte zur Quelle.

Entsprechend dienen `refr_contr` und `read_all_contr` zur Aktualisierung der Daten des Kontrollpultes. Hier findet auch die Untersetzung der Potentiometer -Werte statt.

Die obigen Routinen prüfen selbst nicht die Schreiberlaubnis auf CAMAC, weshalb dies explizit in den aufrufenden Routinen geschehen muß.

**Übertragungsfrequenz eines Datenübertragungszyklus:** Durch Laufzeittests verschiedener Programmteile im Pult-Modus wurde ermittelt, daß bei Auslesen aller 16 Kanäle des Quellen-ADCs innerhalb eines Übertragungszyklus die Wartezeit bis zum Ende der ADC-Konversion etwa 80 % der Übertragungszeit beansprucht. Die restliche Zeit entfällt

zu etwa gleichen Teilen auf die Steuerung des Kontrollpultes und die Übertragung der restlichen Quellenparameter. Da in der Praxis die Einstellung des Strahls durch Beobachtung des Stroms auf einem der Tandem-Cups geschieht, und die Anzeigeinstrumente der Quelle nur eine untergeordnete Rolle spielen, ist die Übertragungsfrequenz der Istwerte von der Quelle von wesentlich geringerer Bedeutung als die Übertragungsfrequenz der Sollwerte zur Quelle. Aus diesem Grunde wurde das Programm so strukturiert, daß im Pult-Modus bei jedem Übertragungszyklus nur jeweils einige Istwerte durch den Quellen-ADC ausgelesen werden, während die Sollwerte komplett übertragen werden. Bei Variation der Anzahl der gelesenen ADC-Kanäle pro Übertragungszyklus ergeben sich folgende Werte:

| ADC-Kanäle / Zyklus | Frequenz der Übertragungszyklen | Frequenz der vollständigen ADC-Übertragung (16 Kanäle) |
|---|---|---|
| 16 | 7.5 Hz | 7.5 Hz |
| 4 | 29 Hz | 7.2 Hz |
| 3 | 35 Hz | 7 Hz |
| 2 | 42 Hz | 5.2 Hz |
| 1 | 58 Hz | 3.6 Hz |

Tabelle 8.2: Mittlere Rate der Übertragunszyklen als Funktion der Anzahl der gelesenen Kanäle des Quellen-ADCs pro Zyklus

Ein dreimaliges Auslesen des ADCs pro CAMAC-Zyklus erscheint als vernünftiger Kompromiß, da dann die Auslesefrequenz noch so hoch ist, daß auf der Anzeige des Kontrollpultes der Eindruck einer kontinuierlichen Veränderung erhalten bleibt, andererseits die Übertragungsrate zur Quelle hoch genug ist, um eine empfindliche Reaktion des Strahls auf die Änderung der Potentiometer zu gewährleisten.

### 8.2.5 Fehlererkennung und Behandlung

#### Serielle Übertragungsfehler

Der serielle Highway-Driver bietet eine Reihe von Möglichkeiten, Übertragungsfehler zu erkennen. Da die Datenübertragung in einem geschlossenen Ring erfolgt, d.h. jedes abgeschickte Datenpaket zum Absender zurückgelangt, wird ein Übertragungsfehler sofort durch den Serial Highway-Driver erkannt und durch verschiedene Bits im Status-Register des SHD angezeigt. Zur Prüfung der Daten enthält jedes Datenpaket eine Reihe von Paritätsbits. Jedes Byte des Paketes enthält ein transversales Paritätsbit, und am Ende des Pakets befindet sich noch ein Byte aus longitudinalen Paritätsbits. Durch Vergleich des Wertes der logischen Exclusiv-Oder-Verknüpfung der einzelnen Bits mit dem Wert des Paritätsbits lassen sich so bei der Übertragung entstandene Fehler im Bitmuster feststellen. Die transversalen Paritätsbits dienen der Überprüfung eines Bytes, die longitudinalen des jeweiligen Bits in allen Bytes des Paketes. Eine Nichtübereinstimmung wird ebenfalls im Statusregister angezeigt.

Die entsprechenden Fehlerbits werden bei jedem übertragenen CAMAC-Befehl kontrolliert, beim Auftreten eines Fehlers wird die Übertragung wiederholt. Die relevanten Fehlermeldungen sind *ADNR, SERR, LPE, TPE, TMO, ERR* (siehe Tabelle 8.3). Auftretende Störungen haben somit lediglich einen negativen Einfluß auf die Übertragungsrate, nicht jedoch auf die Übertragungssicherheit.

Ein automatischer Abbruch des Programms erscheint nur sinnvoll, falls die auftretende Fehlerrate auf den vollständigen Ausfall des Übertragungsweges hindeutet. Beim Auftreten eines Fehlers, der auf die Unterbrechung des Loops hinweist (ADNR und SERR), wird deshalb ein programminterner Zähler `ferrcnt` erhöht. Übersteigt der Zähler einen bestimmten Wert (50), wird das Programm nach Speicherung des aktuellen Zustandes mit einer Fehlermeldung abgebrochen. Damit sich vereinzelt auftretende Übertragungsfehler nicht summieren und zu einem Abbruch des Programmes führen, wird der Zähler `ferrcnt` bei jeder korrekten Übertragung erniedrigt. Ein Programmabbruch erfolgt also nur, falls die Fehlerrate der Übertragung über einen längeren Zeitraum bei über 50 % liegt.

| Bit | Kürzel | Bedeutung |
|---|---|---|
| 1 | $\overline{SQ}$ | Q des letzten seriellen Kommandos war 0 |
| 2 | $\overline{SX}$ | X des letzten seriellen Kommandos war 0 |
| 3 | ERR | Fehler im letzten seriellen Kommando |
| 4 | DERR | Fehler im vorhergehenden seriellen Kommando |
| 5 | ADNR | Crate Adresse des letzten Kommandos wurde nicht erkannt |
| 6 | LPE | Longitudinaler Paritätsfehler in empfangener Message |
| 7 | TMO | Timeout. Keine Antwort nach 350 ms seit dem letzten Kommando |
| 8 | TPE | Transversaler Paritätsfehler in empfangener Message |
| 9 | RDY | Kommandosequenz abgeschlossen |
| 10 | XMR | Senden des letzten Kommandos abgeschlossen |
| 11 | DMD | Demand Message empfangen |
| 12 | SERR | Synchronisation verloren (SYNC LOST) |

Tabelle 8.3: Statusregister des SHD

### Alarmschwellen

Um eine Fehlbedienung der Netzgeräte zu vermeiden, können für die Sollwerte jedes Kanals Ober- und Untergrenzen im Konfigurationsskript festgelegt werden. Die Einstellung eines Sollwertes, der diese Grenzen überschreitet, wird blockiert. Die Überprüfung dieser Grenzen geschieht in der in Abschnitt 8.2.4 beschriebenen Routine `change_values`

Für die Istwerte wurden gleichfalls Alarmschwellen eingebaut, die auch im Konfigurationsskript festgelegt werden und bei Über- bzw. Unterschreitung zur Ausführung eines Shellskripts führen. In diesem Skript können beliebige Aktionen festgelegt werden, z.B. können Mails an Benutzer verschickt oder Kommandos an das Steuerprogramm gesendet werden. Die Überprüfung der Istwerte erfolgt in der Routine `check_alarm`, die in der Timer-*Signal*-Routine aufgerufen wird. Diese überprüft die Alarmschwellen für alle 16 Kanäle. Bei Überschreitung einer Schwelle wird zunächst eine entsprechende Alarmmeldung in ein File geschrieben. Dann wird mit der C-Funktion *fork* ein Child-Prozeß des Steuerprogrammes erzeugt, der mit der Funktion *system* das Shellskript ausführt und anschließend abbricht. Das Steuerprogramm läuft unterdessen weiter. Um zu verhindern, daß das Alarmskript mehrmals ausgeführt wird, wird von der Routine das Flag `Alarmflag` gesetzt, das vor jeder Ausführung der Alarmroutine überprüft wird. Das Flag muß mit dem Kommandozeilenbefehl *resalarm* zurückgesetzt werden.

**Laufzeitfehler des Programmes**

Im Falle eines Fehler-*Signals* , das zum Programmabbruch führt, aber auch im Falle einer beabsichtigten Beendigung des Programmes, wird in die Routine `goodbye` verzweigt. Diese speichert den letzten Zustand der Quelle in die Datei *piq_panic* und sorgt dann durch Rücksetzen des Terminals und Freigabe der Speicherbereiche für eine saubere Beendigung des Programmes. Bei Neustart des Programmes mit der Option *-p* wird das Programm mit den Werten dieser Not-Speicherung initialisiert. Da die Werte auch ohne das Programm in den CAMAC-Bausteinen der Quelle erhalten bleiben, sind so Abbruch und Neustart des Programmes möglich, ohne die Quellenparameter zu beeinflussen.

Ca. alle 3 min. wird - ausgelöst durch das Timer-*Signal* - der Zustand der Quelle in der Datei *piq_values* gespeichert. Diese wird als Initialisierungsfile benutzt, falls das Programm *piq* ohne Zusatzoptionen gestartet wurde. Ein Neustart des Programmes ist auch auf diese Weise problemlos möglich. Ein Fehler während der Laufzeit des Programmes hat also keine Auswirkungen auf die Einstellung der Quelle.

## 8.3 Das Client/Server-Konzept

Auf den Nutzen der Kommunikation anderer Prozesse mit dem Steuerprogramm wurde mehrfach hingewiesen. Auf diese Weise lassen sich komplexere Hilfsprogramme wie grafische Oberflächen entwickeln oder auch aus Meßprogrammen auf anderen Rechnern heraus Befehle an die Quelle schicken.

Das Softwareinterface benutzt die Shared-Memory-Technik und den UNIX *Signal*-Mechanismus. Es hat sich als günstig erwiesen, einen gemeinsamen Kommandozeileninterpreter für direkte Eingabe, Setup-File und das Softwareinterface zu verwenden, um die Befehle zu vereinheitlichen und um das Programm rationeller zu gestalten. Das Interface besteht aus zwei Shared-Memory-Segmenten mit jeweils 255 Byte Länge, einer für den Kommando-String und einer für zurückzugebende Ergebnisse. Das Steuerprogramm signalisiert die Freigabe des Kommandosegmentes durch Schreiben des Wertes 0 in das erste Byte des Segmentes. Ein Client-Prozeß darf erst ein Kommando übertragen, wenn dieses Byte gesetzt ist. Andernfalls könnte es vorkommen, daß eine Kommandoübertragung erfolgt, während das Steuerprogramm noch den vorherigen Befehl abarbeitet, was zum Ignorieren des aktuellen Befehls führen würde. Um den Zugriff mehrerer Client-Prozesse zu ermöglichen, besteht das erste Element des Kommando-Strings aus der PID des aufrufenden Prozesses.

Das Steuerprogramm legt noch zwei weitere Shared-Memory-Segmente an, die Kopien der Datenstrukturen `piq_channel` für die analogen und `piq_status` für die digitalen Quellenparameter enthalten. Mit Hilfe dieser Segmente können Client-Programme auf sämtliche Quellendaten zugreifen, ohne den Betrieb des Steuerprogrammes zu beeinträchtigen.

### 8.3.1 Funktion des Software-Interfaces

Ein Client-Programm, das einen Befehl an das Steuerprogramm übermitteln will, muß zunächst aus dem Lockfile des Steuerprogrammes die Identifikationsnummern der Shared-Memory-Bereiche und die PID des Steuerprogrammes entnehmen und den Zugriff auf die entsprechenden Shared-Memory-Bereiche anmelden. Anschließend muß durch Testen des ersten Bytes des Kommandosegments auf den Wert 0 die Freigabe einer Befehlsübertragung geprüft werden. Nun wird die PID des Prozesses und das auszuführende Kommando in das

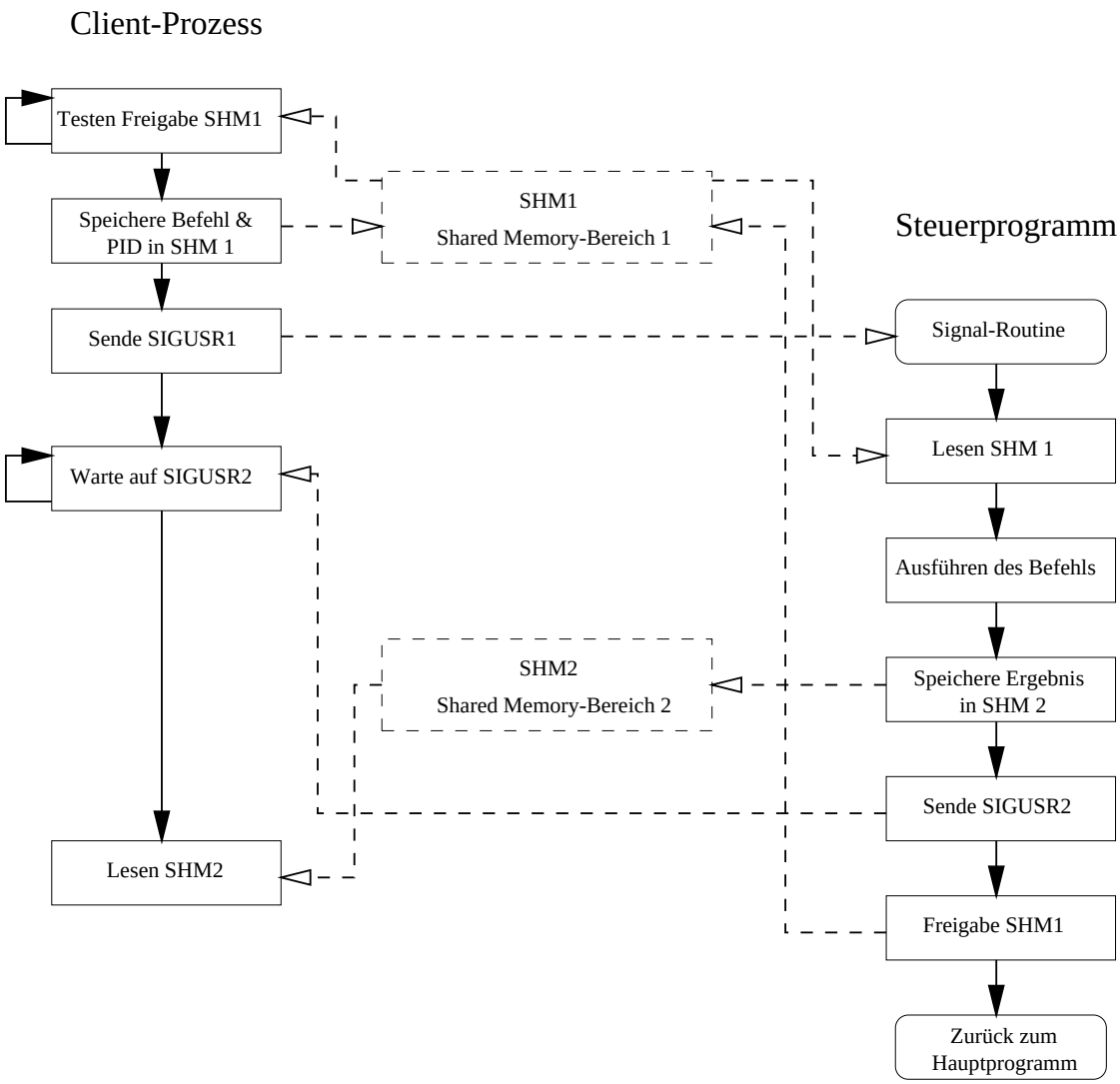


Abbildung 8.5: Softwareinterface

ermittelte Shared-Memory-Segment geschrieben und das *Signal SIGUSR1* an das Steuerprogramm geschickt. Danach geht das Client-Programm mit der C-Funktion *pause* in den Wartezustand.

Das Steuerprogramm verzweigt bei Erhalt von *SIGUSR1* in die Routine `x_irq`, die den String aus dem SHM-Bereich an den internen Befehlsinterpreter übergibt und ausführt. Das Ergebnis des Befehls wird in das Antwortsegment des SHM-Bereichs geschrieben und das *Signal SIGUSR2* an das Client-Programm zurückgeschickt. Dieses befindet sich noch in der *pause*-Funktion, die bei Erhalt des *Signals* beendet wird. Das Ergebnis wird aus dem 2. SHM-Bereich gelesen und ausgewertet. Unterdessen schreibt das Steuerprogramm den Wert 0 in das erste Byte des Kommandosegmentes und gibt damit die Übertragung weiterer Befehle frei.

### 8.3.2 Der Kommandozeileninterpreter

Der Interpreter befindet sich in der Routine `command_line`, der als Parameter ein Befehlsstring sowie ein Flag übergeben wird. Dieses enthält entweder die PID des auslösenden Prozesses oder die Konstante `P_TERMINAL`, die signalisiert, daß das Kommando direkt am

Terminal eingegeben wurde.

Die Kommandos werden im Quelltext in einem Array definiert, das jeweils den Kommandostring, die aufzurufende Prozedur und ein Flag für die Struktur der entsprechenden Parameter enthält. Der Interpreter vergleicht das erste Element des übergebenen Befehlsstrings mit den Kommandostrings aus dem Array. Bei Erfolg werden die entsprechenden Parameter des Befehls aus den übrigen Elementen des Befehlsstrings ermittelt und anschließend in die zugeordnete Routine verzweigt. Die aufgerufenen Routinen schreiben ihr Ergebnis selbständig auf den Bildschirm bzw. in das SHM-Antwort-Segment.

### 8.3.3 Client-Programme

#### piqcmd

Die Aufgabe des Programmes piqcmd besteht darin, den als Parameter übergebenen String an das Steuerprogramm zu senden und das Ergebnis auf der Standardausgabe auszugeben. Es bedient sich dabei der in den letzten Abschnitten beschriebenen Technik. Das Programm dient hauptsächlich der Steuerung der Quelle aus dem Analysator-Meßprogramm. So wurde bei den bisher erfolgten Strahlzeiten aus dem Tcl-Meßprogramm des Analysatorrechners mit einem Remote-Shell-Kommando das Steuerprogramm zunächst mit *piqcmd keymode* auf den Keyboard-Modus gesetzt und danach mit *piqcmd up* bzw. *down* oder *unpol* die Polarisation automatisch umgeschaltet.

```
#include <stdio.h>    /* Include-Dateien */
#include <string.h>
#include <sys/shm.h>
#include <signal.h>
#include <unistd.h>
#include "config.h"
void tu_nix() {} /* Dummy-Funktion */

void wait_for_signal(int wait_signal)
{
  signal(wait_signal,tu_nix); /* Setzen des Signal-Handlers */
  pause();                    /* Warten auf Signal */
}

FILE *Shm_Log;
int shm_rec_id,shm_send_id,shm_chan_id,shm_stat_id;
int master_pid;
char *shm_rec,*shm_send;

int init_shm()  /* Shared-Memory-Bereich initialisieren */
{
  if((Shm_Log=fopen(SHAREFILE,"r"))==NULL) return 1;
  fscanf(Shm_Log,"%d %d %d",&master_pid,
         &shm_send_id,&shm_rec_id);   /* Shm-IDs aus File lesen*/
  fclose(Shm_Log);
  shm_rec=shmat(shm_rec_id,0,0);      /* Shm-Bereiche ermitteln */
  shm_send=shmat(shm_send_id,0,0);
```

```
  return 0;
}
/*
 *   Hauptprogramm
 */
int main(int argc, char **argv)
{
char buffer[255];
int i;
setuid(0);                          /* setuid root */
init_shm();                         /* Shared Memory initialisieren */
buffer[0]=0;
sprintf(buffer,"%d ",getpid());     /* ID ermitteln und in String schreiben*/
for(i=1;i<argc;i++)
  {
    strcat(buffer,argv[i]);
                          /* Kommandozeilenargumente an Bufer haengen */
    strcat(buffer," ");
  }
while(shm_send[0]!=0);              /* Auf Freigabe warten */
strncpy(shm_send,buffer,255);       /* Buffer ins SHM schreiben */
kill((pid_t)master_pid,SIGUSR1);    /* Signal SIGUSR1 an PIQ schreiben */
wait_for_signal(SIGUSR2)            /* Auf Bestaetigung warten */
fprintf(stderr,"%s\n",shm_rec);     /* Ergebnis ausgeben */
}
```

Abbildung 8.6: Implementierung des piqcmd-Programmes

**piqwish**

Dieses Programm ist eine Tcl/Tk-Shell, die um einige Kommandos für die Quellensteuerung erweitert wurde. Das neue Tcl-Kommando `piqsendcmd` sendet nach dem obigen Prinzip sein Argument als Befehl an das Steuerprogramm. Das Ergebnis wird in die Tcl-Variable PiqAnswer kopiert und kann vom Tcl-Programm weiterverarbeitet werden. Es existieren noch weitere Kommandos, die den Quellenstatus aus den die Quellendaten enthaltenden Shared-Memory-Bereichen lesen.

Mit dieser Shell können bequem und schnell grafische Benutzeroberflächen für die Quellensteuerung erstellt werden. Ein Beispiel ist das im Rahmen der Arbeit erstellte Tcl/Tk-Programm *xpiq*.

## 8.4 Sonstige Einrichtungen

### 8.4.1 CAMAC-Server

Das Programm PIQ bietet die Möglichkeit, beliebige CAMAC-Befehle, die über die Kommandozeileneingabe eingegeben oder von anderen Programmen gesendet wurden, über das

CAMAC-Interface zu übertragen. Dazu dient der Befehl 'cam', der als Kommandozeile oder über die Softwareschnittstelle eingegeben werden kann.

Um die externen CAMAC-Befehle mit den internen der Steuerung zu koordinieren, werden die externen Befehle zunächst auf einem FIFO-Stack abgelegt. Zusätzlich wird die PID des aufrufenden Client-Prozesses bzw. bei einem am Terminal eingegebenen Befehl der Wert `TERMINAL_CALL` gespeichert.

Der CAMAC-Stack wird nach jedem Schleifendurchlauf des Hauptprogrammes abgearbeitet. Da ein Schleifendurchlauf im Pult-Modus mit maximaler Frequenz erfolgt und im Keyboard-Modus nur aus einem *pause*-Befehl besteht, der durch das Schreiben des CAMAC-Befehls beendet wird, erfolgt eine Übertragung der externen CAMAC-Kommandos mit einer ähnlich hohen Übertragungsrate wie die der internen CAMAC-Befehle.

Während der Abarbeitung des Befehls wird die CAMAC-Übertragung für andere Routinen durch Setzen des `Block_Camac`-Flags gesperrt. Auf diese Weise ist gewährleistet, daß sich externe CAMAC-Befehle und interner Übertragungszyklus nicht gegenseitig stören. Das Ergebnis des CAMAC-Befehls sowie der Status von Q und X werden in den Ergebnisstring des Shared-Memory-Bereichs geschrieben und an den aufrufenden Prozeß das Handshake-*Signal* gesendet. Anschließend wird der Befehl aus dem Stack entfernt.

### 8.4.2 Werteprotokollierung

Die Steuerung enthält die Möglichkeit, die Quellendaten kontinuierlich zu protokollieren. Zu diesem Zwecke enthält die Datenstruktur `piq_channel` das Flag `dumpflag`, das über Kommandozeilenbefehle manipuliert werden kann. Die Timer-*Signal*-Routine `time_dep` ruft die Routine `dump_perhaps` auf, die mittels eines Zählers den Timer-Takt untersetzt. Bei Ablauf des Zählers erfolgt eine Protokollierung aller Kanäle, bei denen das Flag `dumpflag` gesetzt ist.

Die Werte werden an das bestehende File *piq_dump* angehängt. Das Format ist zeilenorientiert ausgelegt, so daß die Werte leicht mit einem AWK-Skript ausgewertet werden können. Ein Beispiel für ein solches Programm ist unter 10.6 auf Seite 87 beschrieben.

### 8.4.3 Werteübernahme

Die Routine `ist_to_soll` dient der automatischen Übernahme von intern eingestellten Werten auf der Quelle auf die Fernsteuerung. Die Werteübernahme wird mit dem Kommandozeilenbefehl 'overtk' im Programm *piq* aufgerufen.

`ist_to_soll` speichert zunächst die aktuellen Istwerte. Diese werden mit Hilfe der Eichfunktionen in Sollwerte umgerechnet und die neuen Sollwerte an die Quelle übertragen. Ausgenommen sind die Kanäle, die keinen Rückmeldewert besitzen. Diese Kanäle werden in ihrem Zustand belassen.

Nun wird die Routine `auto_ext_dmd` aufgerufen, die zunächst das Bit16-Flag setzt, welches die zusätzliche Übertragung von Bit 16 bei Übertragung der analogen Sollwerte veranlaßt. Dies bewirkt bei den entsprechenden Netzgeräten die Anforderung des externen Betriebszustandes. Dann variiert die Routine bei den Netzgeräten, die einen extern/intern Übernahmemechanismus besitzen, den Bereich $\pm 100 Skt$ rund um den aktuellen Sollwert, bis das Netzgerät auf extern schaltet oder der Bereich ergebnislos abgesucht wurde. Der jeweilige Betriebszustand der Netzgeräte wird mit Hilfe der Routine `get_extint` aus dem Wert des CAMAC-Registers JRE1 ermittelt.

Nach Abarbeitung aller Geräte fordert das Programm den Benutzer auf, die Netzgeräte ohne automatische Werteübernahme auf der Quelle von Hand in den externen Zustand zu schalten.

Nach Bestätigung durch den Benutzer versucht das Programm mit Hilfe der Routine `follow_value`, den Sollwert so zu variieren, daß der ursprüngliche Istwert wieder hergestellt wird. Hiervon sind natürlich ebenfalls die Kanäle ausgeschlossen, die keinen Rückmeldemechanismus besitzen.

Die Eigenschaften der verschiedenen Netzgeräte sind im Feld `flags` der Datenstruktur `piq_channel` gespeichert. Die Eigenschaften der Netzgeräte werden im File *piq_setup* festgelegt. Die dort festgelegten Eigenschaften müssen nicht unbedingt mit den tatsächlichen Eigenschaften der Netzgeräte übereinstimmen. Dadurch ist es möglich, einzelne Netzgeräte bei der automatischen Wertübernahme auszuschließen.

### 8.4.4 Frei konfigurierbare Zusatzbits

Das Programm enthält einige Routinen zur benutzerkonfigurierbaren Erweiterung der Steuerung um sog. Alarmbits. Diese überprüfen die Bits im digitalen Eingaberegister JRS0 auf der Quelle und zeigen gegebenenfalls eine Alarmmeldung an. Die Bits werden über das Setup-File konfiguriert und zusammen mit den Alarmmeldungen bei Initialisierung des Programmes in einer Liste gespeichert.

Die Routine `add_alarmbit(nr,message,flag)` nimmt eine Alarmmeldung in die Liste auf. Der String `message` wird ausgegeben, wenn Bit `nr` im JRE0-Register gleich dem Zustand von Flag (0 oder 1) ist. Diese Routine wird während des Einlesens des Setup-Files aufgerufen.

`free_alarmbit_list` löscht die Liste der Alarmbits und gibt den belegten Speicherplatz frei.

`create_alarmbit_message` erhält als Argument den Inhalt des Registers JRE0, prüft die Bedingungen für Alarmmeldungen und gibt als Ergebnis einen String zurück, der die Meldungen enthält. Diese Routine wird jeweils bei der Ausgabe der digitalen Werte aufgerufen.

# Teil IV

# Erfahrungen mit der Steuerung

# Kapitel 9

# Test der Steuerung und Betriebserfahrung

- Die neue Steuerung wurde während zwei jeweils zweiwöchiger Strahlzeiten mit Erfolg eingesetzt. Nach einigen kleinen Verbesserungen, die während der ersten Tage erfolgten, erwies sie sich als vollkommen zuverlässig.

- Bewährt hat sich die größere Transparenz der Steuerung, da alle Daten protokolliert und explizit sichtbar gemacht werden können. Auf diese Weise ließen sich auch nicht mit der Fernsteuerung zusammenhängende Fehlerursachen wesentlich leichter einkreisen als beim bisherigen System. Anhand des Vergleichs der protokollierten Soll- und Istwerte ist sofort ersichtlich, ob eine Fehlerursache in der Steuerung, den Netzgeräten, dem Pult oder auf der Quelle vorliegt.

- Der Bedienkomfort hat sich gegenüber der alten Steuerung erheblich verbessert, da die neue Steuerung alle Elemente der alten enthält (Potentiometer) und diese um viele zusätzliche Möglichkeiten erweitert. Als äußerst zweckmäßig erwies sich auch das Software-Interface, das eine Umstellung der Polarisation aus dem Meßprogramm heraus ermöglichte. Auch die frei konfigurierbare Untersetzung der Potentiometer erwies sich als nützlich, da eine variable Feinregelung der Quellenparameter erfolgen kann, ohne daß eine Verkleinerung des Regelbereichs auftritt.

- Das Steuerprogramm wurde mehreren Langzeittests unterzogen, wobei es über den Zeitraum von mehreren Tagen stabil arbeitete. Einige kleinere Fehler, die während der Tests auftraten, konnten sofort behoben werden. Der Abbruch und der Neustart des Programmes bei Strahlbetrieb wurden ausführlich getestet, wobei der Strahl sich völlig stabil verhielt.

- Der Datenaustausch zwischen Quelle, Pult und Rechner wurde anhand der Protokollfiles überprüft und arbeitet völlig zuverlässig. Da der Strahl sehr empfindlich auf etwaige abweichende Werte reagiert, konnte auch anhand der Beobachtung des Strahls während der Strahlzeiten die Zuverlässigkeit der Datenübertragung bestätigt werden.

- Die Übertragunssicherheit auf dem Serial Highway wurden mit Hilfe eines Testprogrammes geprüft, das einen Zähler zyklisch erhöhte und dessen Wert über die serielle Leitung in ein Register im Quellencrate schrieb. Der gespeicherte Wert wurde anschließend ausgelesen und mit dem geschriebenen Wert verglichen. Innerhalb einer Laufzeit von 50 h (ca. $10^8$ Zyklen) trat kein Fehler auf. Man kann also davon ausgehen, daß die Datenübertragung zur Quelle zuverlässig arbeitet.

- Die automatische Übernahme der Quellenwerte wurde gründlich getestet und verlief sehr zufriedenstellend. Der Strahlstrom blieb bei Übernahme der auf der Quelle eingestellten Werte auf die Fernsteuerung unverändert. Die Automatisierung stellt einen erheblichen Vorteil gegenüber dem bisherigen System dar.

- Die Steuerung hat sich als außerordentlich stabil gegenüber Hochspannungsüberschlägen des Tandembeschleunigers herausgestellt. Während der zweiten Strahlzeit traten mehrere zum Teil schwere Überschläge auf, die jedoch die Steuerung nicht beeinträchtigten.

  Bei simulierten Ausfällen der Steuerung konnte diese ohne Probleme durch Neustart des Programmes wieder in Gang gesetzt werden.

  Als Fazit kann gesagt werden, daß sich die Betriebssicherheit der Quellen-Fernsteuerung gegenüber dem alten System erheblich erhöht hat.

# Kapitel 10

# Beispiele für Anwendungen

## 10.1 Steuerung per Remote-Shell

Mit Hilfe des Programmes *piqcmd* lassen sich Kommandozeilenbefehle an das Programm *piq* senden. *piqcmd* ist ein Client von *piq* und übergibt diesem seine Argumente über das Softwareinterface als Kommando. Die Rückmeldung von *piq* wird von *piqcmd* in seinen Standardausgabekanal geschrieben.

Man kann *piqcmd* benutzen, um über eine Remote-Shell von anderen Rechnern aus der Quellensteuerung Befehle zu geben, z.B um aus dem Meßprogramm des Analysatorrechners heraus die Polarisationsrichtung umzuschalten. Eine Liste der Kommandozeilenbefehle befindet sich in Anhang D auf Seite 112.

```
rsh piq@ikp.uni-koeln.de piqcmd keymode % Umschalten auf Terminalmodus
rsh piq@ikp.uni-koeln.de piqcmd up      % Schalten Pol auf Up
```

Abbildung 10.1: Beispiel der Umschaltung der Polarisation über eine Remote-Shell. Der Aufruf der Remote-Shell erfolgt mit dem UNIX-Kommando *rsh*, gefolgt von der Adresse des Zielrechners 'piq@ikp.uni-koeln.de'. Danach folgt das auf dem Zielrechner auszuführende Programm *piqcmd*, dem als Parameter das Kommando für die Quellensteuerung 'keymode' bzw. 'up' übergeben wird.

## 10.2 Tcl/Tk-Programmierung mit piqwish

Das Programm *piqwish* ist eine erweiterte Tcl/Tk-Shell, die um Befehle zur Quellensteuerung ergänzt wurde. *piqwish* ist ebenfalls ein Client von *piq*. Die Shell soll hauptsächlich dazu dienen, grafische Benutzeroberflächen für die Steuerung mit Hilfe von Tk zu erstellen. Es sind aber auch andere Anwendungen denkbar, wie z.B. Programme zur automatischen Strahloptimierung. Unter *piqwish* stehen folgende zusätzliche Tk-Befehle zur Verfügung:

- *piqsendcmd* ⟨*Befehl*⟩ schickt sein Argument über das Softwareinterface als Kommando an das Steuerprogramm *piq*. Es erfüllt damit eine ähnliche Funktion wie *piqcmd*, mit dem Unterschied, daß es sich bei *piqcmd* um ein UNIX-Programm handelt, das in UNIX-Shells wie *Bash* oder *Sh* aufgerufen werden kann, während *piqsendcmd* ein Tcl-Kommando ist, das nur in der erweiterten Tcl/Tk-Shell *piqwish* verwendet werden

```
#! /usr/bin/piqwish -f

proc read_adc {ch} {                              ;# Lesen des Kanals ch
                global result                     ;# global deklarieren
                piqsendcmd 7 1 14 $ch 0 0         ;# CAMAC-Lesekommando
                set result [lindex $PiqAnswer 0] ;# Ergebnis ist 1.Elem.
                }                                 ;# des Antwortstrings

button .b -text "Kanal 1" -command "read_adc 1"  ;# Kommandobutton
label .l -textvariable result                     ;# Ergebnisanzeige
pack .b .l                                        ;# Widgets packen
```

Abbildung 10.2: Einfaches Beispiel einer Anwendung von *piqwish*: Dieses Tcl/Tk-Skript erzeugt einen X-Button, der auf Knopfdruck den Wert des Kanals 1 eines zusätzlichen ADCs im Quellencrate liest und das Ergebnis im X-Fenster anzeigt.

kann und speziell der Erstellung von Tcl/Tk-Programmen dient. Nach Abarbeitung des Kommandos steht das Ergebnis in der Tcl-Variablen `PiqAnswer`.

Dem Tcl-Kommando *piqsendcmd* kann jeder dem Kommandozeileninterpreter bekannte Befehl als Argument übergeben werden. Dies macht es möglich, die gesamte Quelle über ein Tcl/Tk-Skript zu steuern. Da die Übermittlung des Befehls durch *Signals* synchronisiert wird, ist gewährleistet, daß das Quellenprogramm unmittelbar auf das Kommando reagiert.

Bei diesem Kommando ist darauf zu achten, daß durch die Synchronisations-*Signals* das Quellenprogramm beeinflußt wird. Obwohl es Befehle zum Erfragen des Quellenstatus gibt, sollten diese nur benutzt werden, wenn der Quellenstatus in Echtzeit abgefragt werden muß. Ansonsten empfielt sich einer der unten genannten Befehle.

- *getpol, getdir, getquench, getmode* ermitteln die entsprechenden Parameter des Quellenstatus und geben diese als Ergebnis zurück. Die Parameter werden nicht durch ein Kommando an das Softwareinterface von *piq* ermittelt, sondern aus einem separaten Shared-Memory-Bereich gelesen, der von piq jeweils mit dem Timerinterrupt geschrieben wird. Das bedeutet, daß durch Benutzung dieser Kommandos das Steuerprogramm im Gegensatz zu piqsendcmd nicht beeinflußt wird. Diese Befehle können also völlig unbedenklich benutzt werden.

- *getchannel* ⟨*channel*⟩ ⟨*field*⟩ ermittelt einen bestimmten Wert von Kanal ⟨*channel*⟩. ⟨*field*⟩ kann dabei folgende Werte haben:

  **n** Name des Kanals
  **ur** Einheit des Ist-Wertes des Kanals
  **uw** Einheit des Soll-Wertes des Kanals
  **ii** Istwert als Integer-Zahl (in Skalenteilen)
  **id** Istwert als Dezimal-Zahl
  **si** Sollwert als Integer-Zahl (in Skalenteilen)
  **sd** Sollwert als Dezimal-Zahl

  Auch dieser Befehl liest seine Werte aus einem speziellen SHM-Segment, es gilt das gleiche wie für die Befehle *getpol* usw.

Mit Hilfe der erweiterten Tcl/Tk-Shell bietet sich ein extrem mächtiges Werkzeug zur Erstellung einzelner Applikationen für die Quelle, von Benutzeroberflächen bis zu Meßprogrammen, Datenaufnahmeprogrammen, Optimierungsprogrammen u.ä. Da auch CAMAC-Befehle über die Tcl-Shell abgesetzt werden können, bietet sich auch die Möglichkeit, zusätzliche Bausteine im Quellencrate über ein Tcl-Programm parallel zum Steuerprozeß zu steuern. Denkbar wäre beispielsweise das Auslesen eines weiteren ADCs zur Überwachung zusätzlicher Meßgrößen auf der Quelle (siehe Abb. 10.2).

## 10.3 Konfiguration

```
### Name ###
name 14 Solenoid
### Einheit ###
unit 14 A
### Eichung ###
scale_rb 14 0.01458
scale_ra 14 0.06
scale_wb 14 0.007071
scale_wa 14 0.116
### Ausgabeformat ###
format 14 %9.1f
### Position in der Bildschirmmaske ###
pos 14 3
### Grenzwerte ###
max 14 2900
min 14 0
#
```

Abbildung 10.3: Ausschnitt aus einem *piq_setup*-File. Angegeben ist jeweils ein Konfigurationsbefehl, gefolgt von der Kanalnummer und dem Wert des zu konfigurierenden Parameters. Der Name von Kanal 14 wird beispielsweise auf 'Solenoid' gesetzt, die Einheit auf 'A' usw.

Die Konfiguration des Programmes erfolgt über die Skripts *camac_setup* und *piq_setup*. *camac_setup* enthält die Adressen der CAMAC-Bausteine und hat folgendes Format:

**shd_branch** Branch-Nr. des Kontrollcrates, in dem sich der serielle Highway-Driver und das IO440-Register befinden

**shd_crate** Crate-Nr. des Kontrollcrates

**shd_station** Station des Seriellen Highway-Drivers

**io440_station** Station des IO440-Registers

**scc_crate** Crate-Nr. des Quellencrates

Das eigentliche Setup-File ist das File *piq_setup*. Hier können alle wichtigen Parameter wie Name und Einheit der Kanäle, Auflösung, Eichung, Alarmschwellen, Ausgabeformat,

Alarmmeldungen usw. eingestellt werden. Das Skript benutzt den Kommandozeileninterpreter, so daß im Prinzip alle Kommandozeilenbefehle möglich sind. Eine Liste aller Befehle befindet sich im Anhang D auf Seite 112.

## 10.4 Alarmeinrichtungen

Für jeden Kanal läßt sich mit den beiden Kommandozeilenbefehlen 'alarmx' bzw. 'alarmn' zwei Alarmschwellen festlegen, bei deren Über-/Unterschreitung das Shellskript */home/knecht3T/etc/piq_alarm* ausgeführt wird. Gleichzeitig wird das File */home/knecht3T/etc/piq_alarm_msg* geschrieben, das die aktuelle Alarmmeldung enthält. Im Skript lassen sich Reaktionen auf den Alarm festlegen, wie z.B. das Senden einer E-Mail an einen bestimmten Benutzer.

```
#! /bin/bash
# Erzeugung einer Alarmmeldung aus Alarm-Message und Statusfile
cat ~/etc/piq_alarm_msg ~/etc/piq_actual > ~/tmp/piq_alarmmail
# Absenden als Mail an Benutzer fj
mail -s "PIQ Alarm!!!" fj@hamlet.ikp.uni-koeln.de < ~/tmp/piq_alarmmail
# Loeschen der Meldung
rm -f /home/fj/tmp/piq_alarmmail
```

Abbildung 10.4: Beispiel für ein Alarmskript

Die freien Bits des JRE10-Eingaberegisters lassen sich ebenfalls für Alarmmeldungen nutzen. Die Konfiguration erfolgt mit dem Kommandozeilenbefehl 'alarmbit' gefolgt von der Nr. des Bits und einer Alarmmeldung. Der Befehl bewirkt, daß bei Setzen des entsprechenden Bits die angegebene Meldung auf dem Schirm ausgegeben wird. Der Befehl 'nalarmbit' bewirkt die Ausgabe der Meldung bei nicht gesetztem Bit.

```
alarmbit  12 <Turbos_Ok>
nalarmbit 12 <Turbos_Alarm>
```

Abbildung 10.5: Konfiguration im File *piq_setup* zur Anzeige des Status der Turbo-Pumpen.

## 10.5 Auswertung des Protokollfiles

Das Format des Protokollfiles piq_dump ist so ausgelegt, daß es sich leicht mit einem AWK-Skript auswerten läßt. AWK ist eine Programmiersprache, die speziell zur Mustererkennung und Auswertung zeilenorientierter Textfiles dient. Die Zeilen eines AWK-Programmes enthalten jeweils ein bestimmtes Muster, das in den Zeilen gesucht wird, und einen Befehl, der ausgeführt wird, wenn das Muster gefunden wurde. Auf die Programmierung im einzelnen soll hier nicht weiter eingegangen werden.

Ein Datensatz im Protokollfile enthält fünf Zeilen, denen jeweils ein Zeilenname vorangestellt ist. Vor jedem Datensatz ist eine Kommentarzeile zur besseren Lesbarkeit des Files

```
#
#  Erstellung Datensatz Zeit[sec]/Istwert[Skt]
#  f"ur Kanal 4 aus piq_dump.
#
BEGIN {c=6}           # c=Kanalnummer +2
BEGIN {t0=818780028}  # Start der Messung bei t0
/Digital/ {t=$3-t0}   # Scannen der Zeit
/Ist-I/ {print t,$c}  # Scannen des Wertes und Ausgabe
```

Abbildung 10.6: Beispiel für ein AWK-Programm, welches aus dem Dump-File ein Wertepaar extrahiert

angefügt, die Nr. des Eintrages und Datum und Uhrzeit enthält. Alle 10 Zeilen ist eine Kommentarzeile mit Namen und Einheit der 16 Kanäle eingefügt.

Die erste Zeile des Datensatzes mit Namen 'Digital' enthält die laufende Nummer des Eintrages, die Zeit in Sekunden [1] , Pol, Dir, Quench. Die anderen 4 Zeilen enthalten jeweils die Werte der 16 Kanäle: Ist-D die Istwerte in dezimaler Form, Ist-I die Istwerte in Integer-Form (Skalenteile), analog Soll-D und Soll-I.

[1] Nach dem UNIX-Zeitformat: angegeben werden die Zahl der Sekunden seit dem 1.1.1970 0:00

# Kapitel 11

# Zusammenfassung

## 11.1 Ergebnisse

Die Ergebnisse der Arbeit lassen sich wie folgt zusammenfassen:

- Ein serieller CAMAC-Highway zur Steuerung der Quelle wurde aufgebaut und getestet. Dabei wurde auch die parallele Optokopplerstrecke der alten Steuerung durch ein serielles System ersetzt. Die serielle Übertragungstechnik ermöglicht aufgrund der erheblich geringeren Kabelkosten eine wesentlich flexiblere Verkabelung des Fernsteuerungssystems. Durch die Möglichkeit der Fehlererkennung in den übertragenen seriellen Datenpaketen wird zudem die Übertragungssicherheit erheblich erhöht.

- Ein PC zur Steuerung der Quelle wurde an CAMAC angebunden. Hierfür wurden Treiberroutinen entwickelt. Auf dem PC wurde das UNIX-kompatible Betriebssystem LINUX installiert. Der PC wurde an das Instituts-LAN angebunden. Diese Maßnahmen ermöglichen die komplette Steuerung und Überwachung der Quelle über das Institutsnetz bzw. Internet. Die vollständige Steuerung der Quelle über den Rechner ermöglicht zudem die Protokollierung der Quellendaten und die Automatisierung von Steuerungsaufgaben. Durch neu geschaffene Visualisierungsmöglichkeiten wird die Fehlersuche auf der Quelle erheblich vereinfacht. Die nun bestehende Verfügbarkeit von CAMAC-Treiberroutinen unter LINUX und die Kostengünstigkeit des Systems aus LINUX-PC und CAMAC-Interface ist zudem richtungweisend für zukünftige Entwicklungen.

- Das Kontrollpult der alten Steuerung wurde unter softwaremäßiger Emulation der alten Steuerplatinen in die neue Steuerung integriert. Durch die alternativen Möglichkeiten der Steuerung über Terminal oder Kontrollpult konnten alle Möglichkeiten des alten Systems in das neue integriert werden. Durch eine beliebig wählbare Untersetzung der Potentiometer des Kontrollpultes und die Automatisierung der Übernahme der auf der Quelle eingestellten Werte wurden zusätzliche Möglichkeiten geschaffen, die den Bedienkomfort erheblich erhöhen. Die softwaremäßige Emulation der Funktionsweise teurer Digital-Potentiometer macht die ursprünglich geplante Entwicklung eines digitalen Kontrollpultes überflüssig.

- Durch die Ersetzung der parallelen Optokopplerstrecke durch eine wartungsfreundliche serielle und die Emulation der Steuerplatinen der Fernbedienungszentrale durch das Steuerprogramm wurden die häufigsten Störungsquellen des alten Fernsteuerungssystems beseitigt.

- Umfangreiche Steuersoftware wurde entwickelt, die die Möglichkeit zur Bedienung über das Terminal, das Kontrollpult oder über externe Programme bietet. Das Programm ist weitestgehend interruptgesteuert, wodurch die benötigte Rechenzeit auf ein Minimum reduziert wird. Über ein Softwareinterface läßt sich auf alle Funktionen der Steuerung aus externen Programmen heraus zugreifen.

- Eine erweiterte Tcl/Tk-Shell zur einfachen Entwicklung und Einbindung grafischer Benutzeroberflächen in das Steuerprogramm wurde entwickelt. Die Shell ermöglicht, schnell und einfach auch komplexe und grafisch aufwendige Dienstprogramme für zusätzliche Steuer- und Visualisierungsaufgaben zu entwickeln, ohne sich mit der inneren Struktur des Steuerprogramms beschäftigen zu müssen. Durch die Möglichkeit, externe CAMAC-Befehle über Tcl/Tk abzusetzen, lassen sich sogar zusätzliche CAMAC-Bausteine problemlos in das Gesamtsystem einbinden und zur Steuerung zusätzlicher Geräte oder erweiterten Datenaufnahme nutzen. Eine grafische Benutzeroberfläche wurde erstellt, die einen Zugriff auf alle Grundfunktionen der Quellensteuerung durch einfache Mausbedienung ermöglicht.

- Das System wurde u.a. während zwei Strahlzeiten ausführlich in der Praxis getestet. Dabei erwies es sich als äußerst zuverlässig und wies gegenüber der alten Steuerung erhebliche Vorteile in der Bedienung und der Betriebssicherheit auf.

## 11.2 Ausblick

Die vorliegende Steuerung ist komplett einsatzfähig. Die im folgenden aufgezählten Aufgaben erhöhen die Wartungs- und Bedienungsfreundlichkeit des Systems, sind aber nicht elementar notwendig.

- Eine lohnenswerte Aufgabe wäre die Modernisierung des CAMAC-Systems auf der Quelle. Hier ist zum einen an den Austausch des ADCs zu denken, da der alte ADC die Übertragungszeit stark verlangsamt.

  Des weiteren könnten die Ausgaberegister für die analogen Werte samt der angeschlossenen DAC-Karten durch eine neu zu entwickelnde CAMAC-Netzgerätekarte ersetzt werden. Bei einer Funktionsstörung der alten Karten treten Probleme auf, da einige Einzelteile der Karten nicht mehr erhältlich sind und die Karten wegen der unterschiedlichen angeschlossenen Netzgeräte oft nicht untereinander austauschbar sind. Hierfür wäre eine universelle CAMAC-Karte nützlich, die eine Speicherung der digitalen Werte vornimmt und deren Ausgangsgröße für die verschiedenartigen Netzgeräte frei konfigurierbar ist, so daß die Karten untereinander ausgetauscht werden könnten. Die Karte könnte ebenfalls einen ADC enthalten, der die digitale Wandlung der Rückgabespannungen übernimmt. Zu beachten ist, daß sie für recht unterschiedliche Ausgangswerte konfigurierbar sein muß, wie $\pm(2..10)V$, Binär, BCD usw.

  Eine Alternative wäre der Umbau aller Netzgeräte auf einen standardisierten Steuereingang. Nachteilig wäre der erhöhte Entwicklungsaufwand und die Inkompatibilität zum alten System.

  Die softwaremäßige Anpassung einer neuen Karte an die Steuerung wäre recht einfach, da wegen der modularen Struktur des Programmes nur einige elementare Befehle wie Lesen und Schreiben der Karte implementiert werden müßten.

- Wünschenswert wäre die saubere Implementierung der CAMAC-Routinen als LINUX Device-Driver. Dies hätte den Vorteil, daß auch fremde Prozesse auf dem PC auf CAMAC zugreifen können, ohne das piq-Programm als Serverprogramm benutzen zu müssen. Des weiteren fiele die Notwendigkeit der Ausführung unter Superuserrechten weg, was einen Vorteil bei der Programmentwicklung und der Systemsicherheit bieten würde.

  Die Entwicklung eines solchen Treibers benötigt eine tiefere Einarbeitung in den LINUX Kernel-Quellcode (gut dokumentiert [Be94]). Die normalen Device-Driver müssen direkt in den Kernel eingebunden, d.h. für jeden Testlauf muß der Kernel neu kompiliert und gebootet werden. Neuere Kernel-Versionen bieten die Möglichkeit, Gerätetreiber als ladbare Module zu verwalten. Dies könnte die Entwicklung vereinfachen, benötigt jedoch eine zusätzliche Einarbeitung in die Programmierung dieser Treiber-Module.

- Die X-Oberfläche ist zwar schon recht komfortabel, die Möglichkeiten und die Benutzerfreundlichkeit von Tcl/Tk verleiten jedoch dazu, weitere Erweiterungen vorzunehmen. So könnte z.B. eine grafische Darstellung der Quellenparameter erfolgen. Möglicher Ausgangspunkt hierfür ist die für die Jülicher CBS entwickelte Oberfläche von Gerhard Krause[Kr96].

  Auch ist die Programmierung kompletter Steuerungsabläufe in Tcl/Tk problemlos möglich, sogar unter Verwendung zusätzlicher CAMAC-Bausteine, die über den im Steuerprogramm enthaltenen CAMAC-Server angesprochen werden können.

  Mögliche Anwendungen wären z.B. die automatische Optimierung der Quellenparameter oder die Aufnahme weiterer Meßwerte auf der Quelle über einen zusätzlichen ADC. Zur automatischen Optimierung ist es erforderlich, die für die Qualität des Strahls maßgebliche Größe $p^2 I$ (Produkt aus Polarisation des Strahls zum Quadrat und Strahlstrom) für den Rechner erfaßbar zu machen.

- Ursprünglich war die Entwicklung eines neuen, digitalen Kontrollpultes im Gespräch. Dieses sollte mittels digitaler Drehgeber anstelle von Potentiometern bedient werden. Hauptvorteil dieses digitalen Pultes sollte die vereinfachte Übernahme der Quellenwerte auf das Pult sein. Diese Aufgabe wurde jedoch schon in der aktuellen Steuerung durch die Umrechnung der Potentiometer -Werte in relative Werte gelöst, so daß ein digitales Pult im Grunde nicht mehr notwendig ist.

# Teil V

# Anhang

# Anhang A

# Betriebsanleitung

## A.1 Übersicht über die Steuerung

- Die Steuerung erfolgt durch einen LINUX-PC mit CAMAC Interface-Karte. Dieser ist am Institutsnetz bzw. Internet angeschlossen und kann von jedem anderen Terminal aus bedient werden. Die Bedienelemente des Kontrollpultes der bisherigen Steuerung sind ebenfalls in die Steuerung integriert, so daß die Quelle wie gewohnt gesteuert werden kann.

- Die Steuersoftware besteht aus dem Programm *piq*, welches sämtliche Steueraufgaben erledigt. Die Bedienung des Programmes erfolgt über Tastatur. Die Ausgabe kann auf einem VT100-fähigen Text-Terminal erfolgen. Zusätzlich existieren das Client-Programm *xpiq*, das eine grafische Benutzeroberfläche erzeugt, und *piqcmd*, über das Kommandos als UNIX-Kommandozeile an die Quellensteuerung gegeben werden können. Ferner existiert das Programm *piqwatch*, das die Online-Überwachung der Quelle erlaubt.

- Es gibt drei Möglichkeiten, das Steuerprogramm *piq* zu bedienen:

  1. Ähnlich wie bisher über das Steuerpult, wobei die absolute Stellung der Potentiometer keine Bedeutung mehr hat, sondern nur die relative Änderung der Position zur bisherigen Stellung. Dies ermöglicht die problemlose Umschaltung zwischen den verschiedenen Steuerungsarten und die automatische Übernahme der auf der Quelle eingestellten Werte. Den Absolutwert eines Potentiometers als aktuellen Sollwert zu übernehmen ist dennoch problemlos möglich.
  2. Durch direkte Tastatursteuerung des Programmes *piq*. Dieses benötigt als Ausgabegerät lediglich ein VT100-Terminal und ist daher auch über einfache Textterminals (Atari mit Terminalemulation) steuerbar. Die Steuerung erfolgt über Tasten oder Kommandozeileneingaben.
  3. Über ein X-Kommandofenster. *xpiq*, ein Client-Programm von *piq*, erzeugt eine grafische Benutzeroberfläche unter X-Windows, über die die Quelle vollständig gesteuert werden kann.

- Die Software läuft in drei verschiedenen Betriebsmodi, zwischen denen über Terminal-Tasten oder das X-Fenster umgeschaltet werden kann:

  **Pult-Modus** Die Bedienung der Steuerung erfolgt über das Kontrollpult.

**Terminal-Modus** Die Bedienung der Steuerung erfolgt über das Terminal (entweder die Tastatur oder das X-Fenster)

**Lock-Modus** Sämtliche Werteänderungen sind gesperrt.

- Die Übernahme von der Quelle kann automatisch erfolgen. Eine Übernahme per Hand ist ebenfalls möglich.(S. Abschnitt A.10)
- Es existieren Möglichkeiten, den aktuellen Quellenwert zu speichern und zu laden, sowie eine Protokollfunktion, die den momentanen Zustand der Quelle in festgelegten zeitlichen Abständen in ein File schreibt.
- Das Programm enthält eine Schnittstelle zur Interprozess-Kommunikation (IPC). Dadurch können verschiedene Client-Programme Befehle an das Steuerprogramm senden und Rückmeldungen von dort empfangen.

## A.2 Die Programme

### piq

Den Hauptteil der Steuerung stellt das Programm *piq* dar, welches sämtliche Steueraufgaben übernimmt. Dieses Programm enthält keine X-Erweiterung und ist in einer VT100-Terminalumgebung lauffähig. Bei Start des Programmes *piq* in einem X-Terminal wird die X-Oberfläche *xpiq* automatisch gestartet.

### xpiq

Auf X-Terminals wird *zusätzlich* das Programm *xpiq* gestartet, welches eine X-Kommandooberfläche erzeugt. Das Programm ist ein Tcl/Tk-Skript für die Shell *piqwish*, einen Tcl/Tk-Interpreter, der um Kommandos zur Steuerung der Quelle erweitert wurde.

*piqwish* ist ein Client von *piq*. *piqwish* und somit *xpiq* sind demzufolge allein nicht lauffähig. *xpiq* kann dennoch allein gestartet werden - Es wartet ca. 5 Minuten, ob *piq* gestartet wurde, und bricht dann ab.

### killpiq

Das Shellskript *killpiq* bricht automatisch alle Prozesse des Steuerprogramms (einschl. *xpiq*) ab. Es wird automatisch bei Beendigung von *piq* gestartet, kann aber auch separat aufgerufen werden.

### piqwatch

Dieses Programm dient zur Online-Überwachung der Quelle über das Netz. Es gibt jede Sekunde den Status der Quelle aus und beansprucht in der übrigen Zeit keine Rechenzeit.

### piqcmd

Das Programm *piqcmd* entstand aus der Notwendigkeit heraus, aus anderen (z.B. Meß-) Programmen heraus die Quelle zu steuern. Das Programm sendet seine Kommandozeilenparameter als Befehl an *piq* und schreibt die Rückmeldung in seine Standardausgabe. So kann von anderen Rechnern bequem über eine Remote-Shell die Quelle gesteuert werden.

## ctrltest

Dieses Testprogramm dient dem Test der Ansteuerung des Kontrollpultes. Per Kommandozeilenoption können die einzelnen Bedienelemente des Kontrollpultes ausgelesen bzw. angesteuert werden. Bei Aufruf des Programmes ohne Kommandozeilenoption erscheint eine kurze Online-Hilfe.

## shtest

*shtest* dient zum Testen des seriellen Highways, d.h. zum Übertragen von Kommandos und der Statusanzeige von SHD und SCC. Es enthält ebenfalls eine Online-Hilfe, die bei Aufruf des Programmes ohne Option angezeigt wird.

## gcam

*gcam* dient der Übertragung von elementaren CAMAC-Kommandos. Eine Online-Hilfe ist enthalten.

## Programmabsturz / Neustart

Bei Beendigung oder Absturz des Programmes bleiben die aktuellen Sollwerte in den CAMAC-Bausteinen der Quelle erhalten, so daß im Prinzip ein problemloser Neustart des Programmes möglich ist. Dem Programm muß lediglich mitgeteilt werden, wie es die Quellenparameter bei einem Neustart initialisieren soll. Folgende Verfahren sind möglich:

- Die sicherste Methode besteht darin, die Quellenparameter regelmäßig in einem Datenfile mit dem Menüpunkt 'file/save' zu speichern. Bei einem Neustart wird das Programm mit der Kommandozeile *piq -f* ⟨*file*⟩ aufgerufen, wobei ⟨file⟩ den Namen des Files symbolisiert.

- Bei einem Programmabbruch oder Absturz versucht das Programm, seine Werte im File */home/knecht3T/etc/piq_panic* zu speichern. Dieses File kann mit der *-p*-Option zur Initialisierung verwendet werden. Dieses Verfahren empfiehlt sich dann, wenn das Programm absichtlich unterbrochen wurde oder die Quellenparameter unmittelbar vor Abbruch des Programmes verändert wurden. Es werden dann die Werte unmittelbar vor Programmabbruch rekonstruiert. Diese Methode ist jedoch nur bei einem manuellen Abbruch des Programmes hundertprozentig sicher, da die Notspeicherung im Falle eines Absturzes versagen kann.

- Ca. alle 5 Minuten wird der Quellenstatus automatisch im File *piq_values* gespeichert. Dieses wird zur Initialisierung verwendet, wenn *piq* ohne Kommandozeilenoption gestartet wurde.

Das Programm legt nach seinem Start das File */home/knecht3T/tmp/piq_share.tmp* an, welches u.a. die Daten für die Shared-Memory-Segmente enthält. Dieses File wird gleichzeitig zur Kontrolle darüber benutzt, ob bereits ein *piq*-Programm gestartet wurde. Sollte es zu einem Programmabsturz gekommen sein, ist es möglich, daß dieses File noch existiert und sich das *piq*-Programm deshalb weigert, zu starten. In diesem Fall sollte das File von Hand mit *rm -f /home/knecht3T/tmp/piq_share.tmp* gelöscht werden.

## A.3 Starten des Systems

1. Einschalten des Kontrollpultes, des Steuercrates, des Quellencrates und der U-Port-Adapter (UPA).

2. Die Konfiguration des Signalweges, die durch die LEDs auf den Frontplatten der U-Port-Adapter angezeigt wird, muß einen geschlossenen Loop ergeben. (Siehe Abb. 4.4) Hierbei sind besonders die SYNC-LEDs an den Eingängen zu beachten, da diese einen einwandfreien Empfang des Signals über die Lichtleiterstrecke anzeigen. Die anderen LEDs signalisieren die Konfiguration des Signalweges. Diese wird im Falle des Slave-UPA aus den Sync-Signalen und dem Loop-Collapsed Signal des Serial Crate-Controllers ermittelt und automatisch eingestellt. Die Konfiguration läßt sich jedoch durch die Dip-Schalter in der Frontplatte manuell einstellen, wobei das obere Feld zwischen automatischer und manueller Einstellung umschaltet und das untere die Konfigurationsparameter angibt. Die Konfiguration des Master-UPA wird allein über den AUX-Port gesteuert. Sie kann mit den in den Stecker eingelassenen Dip-Schaltern Nr.3-6 gewählt werden. Bei Störung des Signalweges ist es oft hilfreich, verschiedene Konfigurationen von Hand auszuprobieren, bevor man nach einem Fehler im Kabel sucht. Oft löst sich das Problem von selbst, oder man erhält wertvolle Hinweise auf die Ursache der Störung.

3. Falls die Sync-Lost LED des seriellen Crate-Controllers leuchtet, liegt ebenfalls eine Störung des Signalweges vor.

4. Nach Einschalten des Rechners wird das Betriebssystem LINUX automatisch gebootet. Es erscheint ein Login-Prompt, unter dem man sich mit User-Namen und Password einloggen kann. X-Windows wird mit dem Kommando *startx* gestartet.

5. Das Steuerprogramm wird mit dem Befehl *piq* gestartet.

6. Nach Starten des Steuerprogrammes muß die Bypassed-LED des seriellen Crate-Controllers erloschen sein. Ansonsten ist dessen Initialisierung fehlgeschlagen, was an einer Störung der Übertragungsstrecke liegen kann.

## A.4 Bedienung der Programme

### Window-Manager

Das auf dem Steuer-PC installierte X-Windows-System benutzt den Window-Manager *fvwm*. Bei diesem Window-Manager wird ein Fenster dadurch aktiviert, daß man mit dem Mauszeiger den oberen Balken des Fensters anklickt. Das Fenster tritt in den Vordergrund und wechselt seine Farbe.

Bei Bedienung des Steuerprogrammes über Tastatur unter dem Fvwm ist vorher das Fenster der Shell zu aktivieren, in der das Steuerprogramm gestartet wurde. Ebenso ist bei Bedienung über *xpiq* das entsprechende Fenster zu aktivieren.

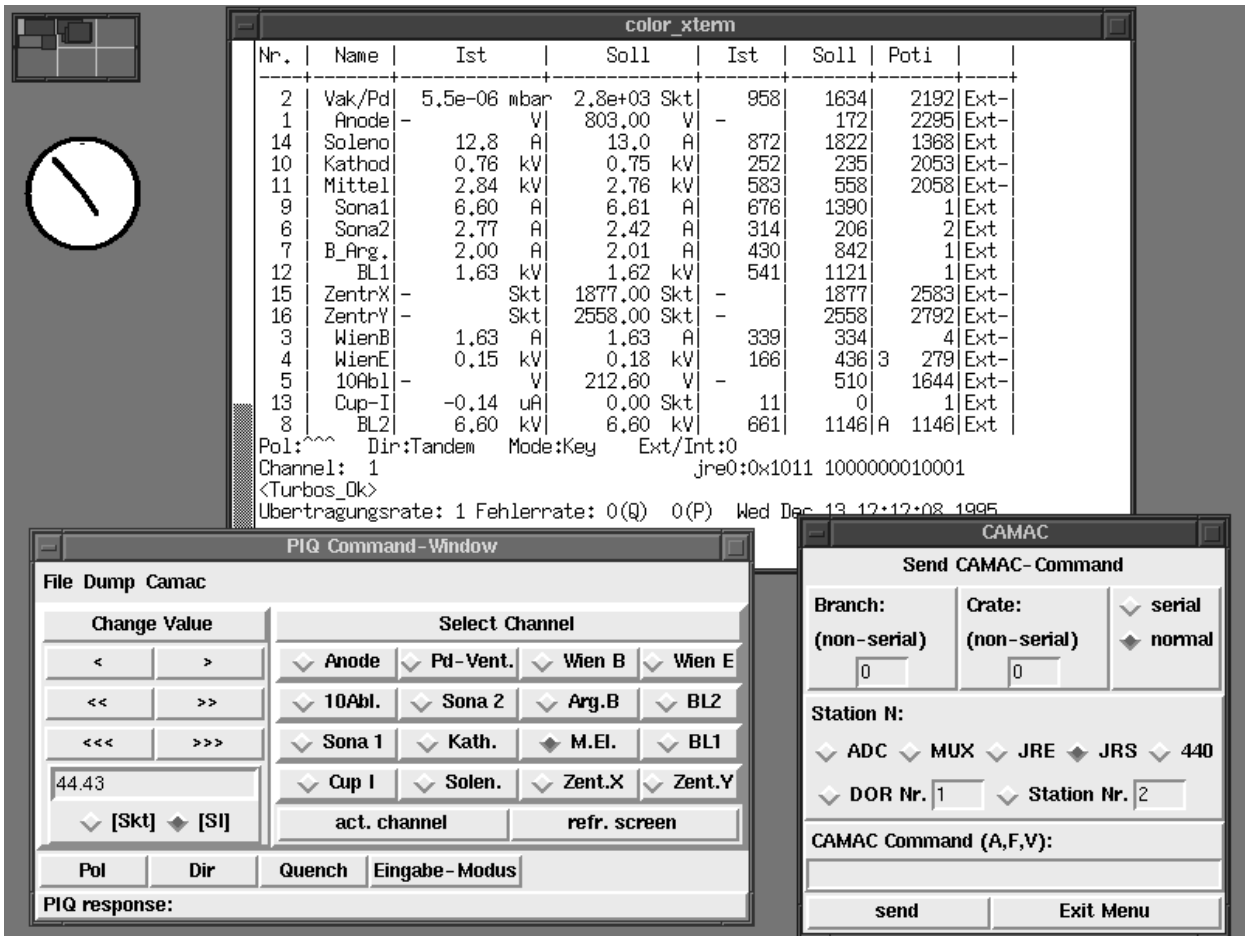
color_xterm
Nr. | Name | Ist | Soll | Ist | Soll | Poti |
2 | Vak/Pd| 5.5e-06 mbar 2.8e+03 Skt| 958| 1634| 2192|Ext-|
1 | Anode|- V| 803.00 V| - | 172| 2295|Ext-|
14 | Soleno| 12.8 A| 13.0 A| 872| 1822| 1368|Ext |
10 | Kathod| 0.76 kV| 0.75 kV| 252| 235| 2053|Ext-|
11 | Mittel| 2.84 kV| 2.76 kV| 583| 558| 2058|Ext-|
9 | Sona1| 6.60 A| 6.61 A| 676| 1390| 1|Ext |
6 | Sona2| 2.77 A| 2.42 A| 314| 206| 2|Ext |
7 | B_Arg.| 2.00 A| 2.01 A| 430| 842| 1|Ext |
12 | BL1| 1.63 kV| 1.62 kV| 541| 1121| 1|Ext |
15 | ZentrX|- Skt| 1877.00 Skt| - | 1877| 2583|Ext-|
16 | ZentrY|- Skt| 2558.00 Skt| - | 2558| 2792|Ext-|
3 | WienB| 1.63 A| 1.63 A| 339| 334| 4|Ext-|
4 | WienE| 0.15 kV| 0.18 kV| 166| 436|3 279|Ext-|
5 | 10Abl|- V| 212.60 V| - | 510| 1644|Ext-|
13 | Cup-I| -0.14 uA| 0.00 Skt| 11| 0| 1|Ext |
8 | BL2| 6.60 kV| 6.60 kV| 661| 1146|A 1146|Ext |
Pol:^^^ Dir:Tandem Mode:Key Ext/Int:0
Channel: 1 jre0:0x1011 1000000010001
<Turbos_Ok>
Ubertragungsrate: 1 Fehlerrate: 0(Q) 0(P) Wed Dec 13 12:12:08 1995
PIQ Command-Window
File Dump Camac
Change Value
Select Channel
< >
<< >>
<<< >>>
44.43
[Skt] [SI]
Anode Pd-Vent. Wien B Wien E
10Abl. Sona 2 Arg.B BL2
Sona 1 Kath. M.El. BL1
Cup I Solen. Zent.X Zent.Y
act. channel refr. screen
Pol Dir Quench Eingabe-Modus
PIQ response:
CAMAC
Send CAMAC-Command
Branch: (non-serial) 0
Crate: (non-serial) 0
serial
normal
Station N:
ADC MUX JRE JRS 440
DOR Nr. 1 Station Nr. 2
CAMAC Command (A,F,V):
send
Exit Menu

Abbildung A.1: Screenshot des Steuerprogrammes.

### Die drei Betriebsmodi

Die Steuerung kann in drei Betriebszuständen laufen:

- **Pult-Modus** In dieser Betriebsart erfolgt die Bedienung über das Kontrollpult. Werteänderungen über das Terminal werden nicht registriert. Eine Übertragung von und zur Quelle bzw. zum Pult erfolgt kontinuierlich, wodurch das Programm sehr viel Rechenzeit in Anspruch nimmt.

- **Terminal-Modus** Alle Eingaben erfolgen über das Terminal, entweder durch Betätigung einzelner Tasten, durch Eingabe einer Kommandozeile oder durch das X-Kommandofenster. Datenübertragung erfolgt nur bei einer Werteänderung sowie in bestimmten zeitlichen Intervallen (z.Zt. eingestellt [1] auf etwa 0.5 sec) , so daß dieser Modus entschieden weniger Rechenzeit in Anspruch nimmt.

- **Lock-Modus** Dieser Modus entspricht dem Terminal-Modus, jedoch sind hier alle Werteänderungen gesperrt.

### Umschalten zwischen den Modi

Auf der Tastatur wird zwischen den drei Modi durch die Tasten 'k' (Keyboard- bzw. Terminal-Modus) , 'p' (Pult-Modus) und 'l' (Lock-Modus) umgeschaltet.

Im X-Kommandofenster kann der Modus im Menü 'Eingabe-Modus' durch Anklicken geändert werden. Der aktuelle Modus wird in der Bildschirm-Statuszeile angezeigt.

*Das Programm ist so ausgelegt, daß beim Umschalten der Betriebsmodi eine automatische Umkalibrierung der Potentiometer erfolgt. Eine Umschaltung zwischen den Modi ist so völlig gefahrlos, der aktuelle Sollwert bleibt auf alle Fälle erhalten.*

### Eingabe von Kommandozeilen

Viele Funktionen des Programmes lassen sich über Kommandozeilenbefehle abrufen. Diese Befehle können einerseits von Client-Programmen an das Steuerprogramm gesendet werden, andererseits aber auch direkt im Terminal-Fenster des Steuerprogrammes eingegeben werden. Zur Eingabe einer Kommandozeile drückt man zunächst die Taste 'i' (Input). Es erscheint ein Prompt, hinter dem man die Zeile eingeben und durch Return abschließen kann.

## A.5 Bildschirmanzeige

Die Bildschirmausgabe erfolgt nach jedem Befehl , der eine Werteänderung auf der Quelle verursacht (Keyboard- oder X-Kommando), sowie bei jedem Übertragungs-Zeittakt. (Das gilt auch für den Pult-Modus, da eine kontinuierliche Bildschirmausgabe das System stark verlangsamt.)

Im oberen Bildschirmbereich werden die Soll- und Istwerte der entsprechenden Kanäle sowohl in Skalenteilen als auch in physikalischen Einheiten angezeigt. Die Werte in physikalischen Einheiten beruhen auf einer linearen Eichung, deren Parameter im Setup-File

[1] Der Wert des Zeitintervalls läßt sich in der Datei *piq_setup* mit dem Befehl 'refresh_time' festlegen.

festgelegt wurden (siehe unten). Ebenfalls angezeigt wird der extern/intern-Status des jeweiligen Kanals. Ein '-' hinter dem jeweiligen Wert gibt an, daß das jeweilige Netzgerät keinen Extern/Intern Übernahmemechanismus besitzt.

In einer weiteren Spalte werden die Potentiometer-Absolutwerte angezeigt. Dies ist hauptsächlich für die Fehlersuche von Bedeutung. Eine Untersetzung der Potentiometer wird durch eine Zahl am linken Rand der Spalte dargestellt, die die Untersetzung als Exponent einer Zweierpotenz angibt. Falls das Potentiometer im Absolutmodus arbeitet, erscheint an dieser Stelle ein 'A'.

Im unteren Bildschirmbereich werden die digitalen Parameter der Quelle (Polarisation, Cup/Tandem und Quench), der Eingabemodus (Key, Pult, Lock) und die Anforderung einer Werteübernahme auf der Quelle (Ext: 0 bzw. 1) angezeigt.

Die Werte der digitalen Parameter stellen dabei die tatsächlichen Werte, nicht die Sollwerte dar. Ausnahme bildet hier 'Quench all', das keinen Rückmeldewert besitzt. Da die Umschaltung auf der Quelle meist kurz hinter der Anforderung erfolgt, ist eine Anzeige der Sollwerte nicht notwendig.

In der untersten Bildschirmzeile wird der aktuelle Kanal angezeigt, d.h. der Kanal, auf den die **Tastaturkommandos** wirken. Die durch das X-Fenster gegebenen Kommandos wirken nicht auf diesen Kanal, sondern auf den im X-Fenster selektierten. [2]

Zusätzlich angezeigt werden die aktuelle Übertragungsrate, die Rate der aufgetretenen Übertragungsfehler auf dem Serial Highway und die Rate der Kontrollpultfehler (s. Anhang B). Die Raten beziehen sich auf die Zeitdauer zwischen zwei Auffrischungs-Zeittakten des Programms, sie sind also nur ein relatives Maß und nicht in der Einheit 'Hz' zu verstehen.

## A.6 Bedienung über das Kontrollpult

Das Kontrollpult ermöglicht die Bedienung der Quelle über Potentiometer und Zeigerinstrumente wie bei der bisherigen Steuerung. Dies ist äußerst nützlich bei der Optimierung des Strahls und verlangt vom Benutzer nur eine minimale Umstellung.

Zur Steuerung über das Pult muß lediglich am Terminal der Pult-Modus durch Drücken der Taste 'p' oder Anklicken des Menüpunktes im X-Fenster selektiert werden, danach kann die Quelle über die Potentiometer und Schalter des Pultes gesteuert werden.

### Analoge Werte

Ein wesentlicher Unterschied zur alten Steuerung ist, daß die Stellung der Potentiometer keine absolute Bedeutung mehr hat, sondern nur die relative Werteänderung registriert und verarbeitet wird. Auf diese Weise ist ein problemloses Umschalten zwischen den Betriebsmodi möglich. **Die Anzeigeinstrumente des Kontrollpultes zeigen jedoch nach wie vor die Absolutwerte an!** *(siehe auch Abschnitt A.10)*

Durch Setzen des Flags 'poti_abs' mit Hilfe des Kommandozeilenbefehls 'flag' wird erreicht, daß der Wert des Potentiometers weiterhin absolut interpretiert wird. Bei Umschalten zwischen Pult- und Terminalmodus kann es in diesem Fall jedoch zu sprunghaften Änderungen des Sollwertes kommen. Der Absolutmodus eines Potentiometers wird in der Bildschirmmaske durch ein 'A' vor dem Potentiometerwert angezeigt.

[2] Eine Übernahme des im X-Fenster selektierten Kanals kann durch den Kommandobutton 'act. chan.' im X-Fenster erfolgen.

### Rücksetzen der Potentiometer-Skala

Sollte das Problem auftreten, daß bei Bedienung der Potentiometer das Ende des Drehbereichs erreicht wird, kann einfach durch Umschalten auf den Lock-Modus die Werteänderung blockiert werden. Anschließend kann das Potentiometer nach Belieben zurückgedreht werden. Bei erneuter Umschaltung auf den Pult-Modus wird das Potentiometer automatisch auf den aktuellen Sollwert kalibriert, so daß dieser erhalten bleibt.

### Untersetzung der Potentiometer

Da der Strahl sehr empfindlich auf einige Quellenparameter reagiert, hat sich eine Untersetzung der Potentiometer-Skala als vorteilhaft erwiesen. Die Untersetzung des Potentiometers wird mit dem Kommando 'res $\langle nr \rangle \langle bit \rangle$' festgelegt. $\langle nr \rangle$ gibt die Nummer des zu untersetzenden Kanals an, $\langle bit \rangle$ die Anzahl der Bits, um die der Potentiometer -Wert nach rechts verschoben werden soll. Die Verschiebung um 1 Bit entspricht einer Untersetzung um den Faktor 2. Die Anzahl der Bits wird links neben den Potentiometerwerten im Ausgabefenster des Steuerprogrammes angezeigt.

### Änderung der digitalen Werte

Über die Kippschalter des Pultes lassen sich Polarisation, Cup/Tandem und Quench all umschalten. Die Istwerte für Polarisation und Cup/Tandem werden auf den LEDs angezeigt. Zu beachten ist, daß die Werte der Kippschalter immer noch absolute Bedeutung haben, so daß bei Umschaltung auf Pult-Modus die am Terminal eingestellten digitalen Soll-Werte nicht erhalten bleiben, sondern die auf den Schaltern eingestellten Werte übernommen werden. Beim Schalten von Pult- auf Keyboard-Modus bleiben jedoch die Sollwerte erhalten.

## A.7 Bedienung über die Tastatur

Vor der Werteänderung über die Tastatur muß sich das Programm im Terminal-Modus befinden, der sich durch Drücken der Taste 'k' (für Keyboard) oder im Menü 'Eingabemodus' des X-Fensters selektieren läßt.

### Selektierung des aktuellen Kanals

Die Tastaturkommandos wirken jeweils nur auf einen Kanal, der zuvor selektiert werden muß. Die Auswahl dieses Kanals geschieht durch zweistellige Eingabe über die Tastatur. Die Eingabe erfolgt zyklisch, es ist keine Bestätigungstaste wie Return erforderlich. Die Eingabe '0 1' wählt also Kanal 1 aus, die weitere Eingabe einer '2' Kanal 12. Unsinnige Eingaben werden vom Programm erkannt, die Eingabe '5 3' wählt Kanal 3 und nicht 53 aus. Die selektierte Kanalnummer wird im unteren Teil des Bildschirmes angezeigt.

### Analoge Werte

Folgende Tasten ändern den Wert des aktuellen Kanals:

**+/-** Werteänderung um $+/-1$
**w/z** Werteänderung um $+/-10$
**e/x** Werteänderung um $+/-100$

| Nr. | +/− | Nr. | +/− | Nr. | +/− | Nr. | +/− |
|---|---|---|---|---|---|---|---|
| 1 | 1/q | 5 | 5/t | 9 | a/y | 13 | g/b |
| 2 | 2/w | 6 | 6/z | 10 | s/x | 14 | h/n |
| 3 | 3/e | 7 | 7/u | 11 | d/c | 15 | j/m |
| 4 | 4/r | 8 | 8/i | 12 | f/v | 16 | k/, |

Tabelle A.1: Direktbelegung der Tastatur

### Digitale Werte

**u/d/n** Umschalten Polarisation: Up/Down/Unpol
**c/t** Umschalten Strahlrichtung: Cup/Tandem
bf a Quench-all umschalten

### Weitere Tastaturkommandos

**l/k/p** Modus-Umschaltung: Lock , Keyboard (=Terminal) und Pultmodus
**r** Refresh Screen: Der Bildschirminhalt wird aktualisiert.
**z** Zap Bit 16: Anforderung des externen Modus von den Netzgeräten der Steuerung.
**i** Eingabe einer Kommandozeile.
**s** Aktuellen Status in Protokollfile schreiben.

### Alternative Tastaturbelegung

Durch die Taste ']' kann auf eine andere Tastaturbelegung geschaltet werden, in der jedem Kanal zwei Tasten zugeordnet sind, die den Wert um +/- 1 erhöhen. Auf dem Bildschirm erscheint dann anstelle der aktuellen Kanalnummer die Meldung 'Direct mode'. Zurückgeschaltet wird wieder mit ']'.

## A.8 Bedienung über das X-Fenster

Das Zusatzprogramm *xpiq* erzeugt unter X-Windows ein Kommandofenster, über das die Quelle bequem mit der Maus gesteuert werden kann. Für die Änderung der Werte ist es ebenfalls nötig, daß sich das Programm *piq* im Terminal-Modus befindet, der im Menüpunkt 'Eingabemodus' eingestellt werden kann.

### Auswahl des aktuellen Kanals

Zunächst ist es auch hier nötig, den Kanal zu selektieren, auf den die Eingaben wirken sollen. Dies geschieht im rechten Feld des Kommandofensters durch Anklicken des Buttons mit dem entsprechenden Kanalnamen. Der hier selektierte Kanal wird nicht automatisch als der aktuelle Kanal für die Tastatureingabe übernommen; dies kann jedoch durch Drücken des Buttons 'Actual Channel' unterhalb des Feldes zur Kanalselektion erreicht werden.

### Analoge Werte

Die Änderung der analogen Werte erfolgt über die 6 linken Buttons mit den '⟩' bzw. '⟨' -Symbolen um jeweils 1,10 und 100 Skalenteile.

Der Wert kann ebenfalls absolut eingegeben werden, und zwar über das 'entry'-Feld unterhalb der Werteänderungsbuttons. Die Radiobuttons '[Skt]' und '[SI]' legen dabei fest, ob der Wert als Skalenteile oder in physikalischen Einheiten zu verstehen ist. Die Übertragung des Wertes erfolgt nach Drücken der Return-Taste.

### Digitale Werte

Eine Änderung der Polarisationsrichtung erfolgt über das Menü 'Pol'. Im Menü 'Dir' wird die Strahlrichtung (Cup oder Tandem) gewählt. Der Quench-all-Zustand kann im Menü 'Quench' umgeschaltet werden. Der aktuelle Zustand wird neben den Buttons dargestellt. Eine Änderung des Eingabemodus kann im gleichnamigen Menü erfolgen. Hier kann auch Bit 16 zur Anforderung des externen Bedienstatus der Netzgeräte angefordert werden.

## A.9 Bedienung über Kommandozeilen

Das Programm *piq* enthält einen Kommandozeileninterpreter, über den sich alle Funktionen des Programmes steuern lassen. Diesem Kommandozeileninterpreter lassen sich auf verschiedene Weise Kommandos übergeben:

- Direkteingabe im Terminalfenster des Steuerprogrammes. Hierzu drückt man Taste i. Es erscheint eine Eingabeaufforderung, in die das Kommando eingegeben werden und mit Return abgeschlossen werden kann. Eventuell muß zur Bestätigung anschließend noch einmal Return gedrückt werden.

- Die Kommandos des Files piq_setup, das nach Start des Programmes gelesen wird, werden durch den Kommandozeileninterpreter ausgeführt.

- Über Hilfsprogramme durch das Software-Interface. Von dieser Technik macht das Programm piqcmd und die Tcl/Tk-Shell piqwish Gebrauch. Näheres in Abschnitt 10.2.

Folgende Kommandozeilenbefehle existieren zur Werteänderung:

| | |
|---|---|
| **set k d** | Setzt Kanal k auf Wert d (Phys. Einheiten) |
| **iset k n** | Setzt Kanal k auf n Skalenteile |
| **add k n** | Erhöht Kanal k um n Skalenteile (n kann auch negativ sein) |
| **up/down/unpol** | Umschaltung der Polarisation |
| **cup/tandem** | Umschaltung der Strahlrichtung |
| **qall/qoff** | Quench all on/off |

Eine Übersicht aller Kommandos befindet sich in Anhang D.

## A.10 Werteübernahme von der Quelle

Die Netzgeräte auf der Quelle lassen sich manuell zwischen einem externen (ferngesteuerten) und internen Betriebsmodus umschalten. Bei Erzeugung eines Strahls werden zunächst die entsprechenden Parameter direkt auf der Quelle im internen Modus der Netzgeräte eingestellt und diese erst danach auf den externen Modus umgeschaltet. Da die auf der Fernsteuerung eingestellten Werte nicht notwendigerweise den intern eingestellten Werten entspechen, besitzen die älteren Netzgeräte der Fernsteuerung einen Wertnachführmechanismus: Fordert man für diese Netzgeräte mit Hilfe des auf den Netzgeräten angebrachten Schalters den externen Betriebszustand an, verbleiben sie zunächst im internen Zustand. Die Umschaltung in den externen Zustand erfolgt erst, wenn dieser von der Fernsteuerung mit Setzen von Bit 16 bei der Übertragung des Sollwertes angefordert wird und der extern angeforderte Wert mit dem intern eingestellten in etwa übereinstimmt. Die neueren Netzgeräte besitzen diesen Mechanismus nicht.

Das Steuerprogramm enthält die Möglichkeit, den Vorgang der Werteübernahme zu automatisieren. Zusätzlich erfolgt eine Übernahme der Werte der Netzgeräte ohne Werteübernahmemechanismus. Das Programm versucht außerdem, die intern eingestellten Istwerte nach erfolgter Übernahme so gut wie möglich zu restaurieren. Voraussetzung ist natürlich, daß der Istwert zur Fernsteuerung zurückgemeldet wird, was bei einigen Geräten nicht der Fall ist.

### Automatische Übernahme

Die automatische Übernahmeroutine rechnet zunächst den Ist-Wert aller Netzgeräte, die einen Rückmeldewert besitzen, mit Hilfe der im Konfigurationsfile festgelegten Eichparameter in einen Sollwert um und schickt diesen zur Quelle. Dann wird bei den Geräten mit automatischem Nachführungsmechanismus der Sollwert so lange variiert, bis diese auf 'extern' schalten oder den vorgegebenen Wertebereich ($\pm 100 Skt$) überschreiten. Im letzteren Falle wird eine Fehlermeldung ausgegeben. Anschließend wird durch Variation des Sollwertes versucht, den ursprünglichen Istwert wieder herzustellen.

Ein Übernahmevorgang hat also so zu erfolgen:

1. Einstellen der Netzgeräte intern auf der Quelle.

2. Netzgeräte mit Nachführmechanismus auf 'extern' schalten, Geräte ohne Nachführmechanismus müssen auf 'intern' stehen bleiben, damit sie den intern eingestellten Wert halten.

3. Werteübernahme durch Eingabe der Kommandozeile 'tkover' (nach Drücken der Taste i) ausführen.

4. Das Programm hält nach einiger Zeit an mit der Aufforderung, die Geräte auf externen Betriebszustand zu schalten.

5. Alle Netzgeräte auf der Quelle auf extern schalten.

6. 'ok' zur Bestätigung im Programm eingeben.

7. Das Programm restauriert nun die alten Istwerte und geht danach wieder in den normalen Betriebszustand über.

## Manuelle Werteübernahme

Die Steuerung fordert bei der Quelle den externen Betriebszustand an, indem sie bei der CAMAC-Übertragung der analogen Sollwerte Bit 16 setzt. Dies kann durch ein Flag im Steuerprogramm erzwungen werden. Umschaltung des Flags geschieht im X-Fenster durch Selektieren des Menüpunktes 'Eingabemodus/Extern anfordern'oder durch die Taste 'z' (zap flag). Angezeigt wird das Flag in der Statuszeile unter 'Ext:'.

Ist das Flag gesetzt, wird Bit 16 bei jeder Übertragung zur Quelle gesetzt. Die manuelle Werteübernahme kann also ganz nach Belieben über X-Fenster, Tastatur oder Potentiometer erfolgen.

## Übernahme eines Potentiometer-Absolutwertes

Die Stellung der Potentiometer hat in der neuen Steuerung nur noch relative Bedeutung. Soll der Absolutwert eines Potentiometers als Sollwert übernommen werden, etwa weil man sich die Einstellung aus einer früheren Messung gemerkt hat, dient dazu die Kommandozeilenfunktion 'getspoti ⟨nr⟩' . Die Funktion kann ebenfalls über einen Menüpunkt im X-Fenster angewählt werden, wobei als Kanal der im X-Fenster selektierte Kanal angenommen wird. Über die Funktion 'getpoti' werden die Werte *aller* Potentiometer übernommen. Hierfür existiert ebenfalls ein Menüpunkt im X-Fenster.

Die Übernahme des Absolutwertes stellt einen einmaligen Vorgang dar, danach arbeitet das Potentiometer wieder 'relativ'.

## Protokollfunktion

Um die Werte der Quelle während des Betriebes zu protokollieren, existieren eine Reihe von Funktionen, die sowohl als Kommandozeile als auch im X-Fenster aufgerufen werden können. Die Protokollierung ist standardmäßig für alle Kanäle eingeschaltet und auf ein Intervall von ca. 5 min gesetzt, so daß man sich normalerweise um diesen Punkt nicht zu kümmern braucht.

Jeder Kanal besitzt ein Flag, das festlegt, ob er in das Protokoll aufgenommen werden soll oder nicht. Die zu protokollierenden Kanäle können über ein ASCII-File (Format: durch Space getrennte Integer-Zahlen) im Menüpunkt 'dump/load channel list' eingelesen werden. Einzelne Kanäle können im Menü 'dump/Optionen' durch Eingabe des Kanals und Betätigung des Buttons 'add' bzw. 'rem' entfernt oder hinzugefügt werden. Hier wird auch das Zeitintervall angegeben, nach dem eine Protokollierung erfolgen soll. Die Liste aller protokollierten Kanäle wird durch 'dump/Show list' ausgegeben. Die Protokollierung wird durch 'dump/start' bzw. 'stop' gestartet und angehalten.

Für alle Funktionen im X-Fenster existieren äquivalente Befehle. Die Flags der Kanäle können durch 'dadd ⟨nr⟩' bzw. 'drem ⟨nr⟩ gesetzt bzw. gelöscht werden. Durch 'dlist ⟨file⟩' wird die Liste geladen, durch 'dshow' angezeigt. 'dstart t' startet das Protokoll mit einem Zeitintervall von t Sekunden, 'dend' hält die Protokollierung an.

Die Werte werden nach jedem Zeitintervall mit Datum, Uhrzeit, Ist- und Sollwerten, geeicht und ungeeicht, an das File *piq_dump* angehängt. Zur ständigen Anzeige des Files kann das UNIX-Kommando *tail -f /home/knecht3T/etc/piq_dump* verwendet werden.

Zur Erzwingung eines Eintrages kann die Taste 's' betätigt werden, die den aktuellen Status sofort in das Protokollfile schreibt.

## A.11 CAMAC-Kommandos

Das Programm *piq* ist in der Lage, beliebige CAMAC-Kommandos an das angeschlossene Steuercrate und über den Serial Highway zu senden. Die Übertragung der Kommandos findet zwischen zwei Übertragungen zur Quelle statt. Auf diese Weise ist eine Havarie mit den CAMAC-Kommandos zur Quellensteuerung ausgeschlossen. Kommandos, die während des Übertragungszyklus an *piq* gegeben werden, werden in eine Warteschlange gestellt und zu gegebener Zeit ausgeführt. Auf diese Weise ist gewährleistet, daß kein Kommando verloren geht und auch komplexere Programme auf *piq* als CAMAC-Server zugreifen können.

Die CAMAC-Kommandos werden direkt oder über das Software-Interface als Kommandozeile mit der Syntax $cam < branch >< crate >< station >< a >< f > [value]$ an *piq* übergeben. Als Ergebnis erhält man einen String mit dem zurückgegebenen Wert sowie den Zustand von Q und X zurück ('Q'/'X' , falls das entsprechende Flag gesetzt ist, sonst '-'). Der Branch 7 bedeutet, daß das Kommando über den seriellen Highway-Driver geschickt wird.

Zur komfortablen Eingabe steht ein spezielles Eingabefenster im Programm *xpiq* zur Verfügung. Dieses wird in der oberen Menüzeile im Menüpunkt 'camac/Send CAMAC Command' angewählt. Im oberen Block kann die Branch- und Crateadresse des CAMAC-Befehls eingegeben werden und mit zwei Buttons die direkte Absendung des Befehls oder das Senden über den seriellen Highway ausgewählt werden. Bei letzterem werden eingestellte Branch- und Crateadresse ignoriert und der Befehl an das Quellencrate geschickt.

Im Menü darunter läßt sich die Stations-Nr. einstellen: Entweder eins der vordefinierten Quellenmodule (bei den dualen Ausgaberegistern muß zusätzlich die Nr. 1-8 angegeben werden) oder eine frei definierte Adresse.

In der unteren Eingabezeile wird schließlich das CAMAC-Kommando, bestehend aus Subadresse, Funktion und ggf. Wert, durch Kommata getrennt, eingegeben.

Das Kommando wird mit dem 'send'-Button abgeschickt. Das Eingabemenü wird mit dem 'exit'-Button verlassen.

## A.12 Files

Die Setup-Files für das Programm *piq* befinden sich in */home/knecht3T/etc*.

### piq_setup

Dieses File enthält die Grundeinstellungen des Programmes für die Quelle, wie Kanalname, Eichparameter usw. Die Befehle werden durch den Kommandozeileninterpreter ausgeführt. Syntax und Bedeutung der Befehle sind im Anhang beschrieben.

### camac_setup

Dieses File enthält die CAMAC-Adressen des Serial Highway-Drivers (shd_branch, shd_crate, shd_station) und des IO440-Registers(io440_station) sowie die Crate-Nummer des Quellencrates (scc_crate).

### piq_values

Ein Datenfile, welches ca. alle 5 min. automatisch aktualisiert und bei Start des Programmes zur Initialisierung benutzt wird. Die Parameter:

| | |
|---|---|
| **dir** | 0=Tandem, 1=Cup |
| **pol** | 0=Unpol, 1=Up, 2=down |
| **quench** | 0=off, 1=Quench all |
| **disp** ⟨**channel**⟩ ⟨**value**⟩ | Wert des Displays (Istwert) in Skt |
| **poti** ⟨**channel**⟩ ⟨**value**⟩ | Wert des Potentiometers in Skt |
| **key** ⟨**channel**⟩ ⟨**value**⟩ | Wert des Sollwertes in Skt |

### piq_share.tmp

Dieses File in '/home/knecht3T/tmp' dient zur Übermittlung der Shared-Memory-Bereiche und der PID an das Client-Programm. Es wird außerdem als Lockfile verwendet, um zu verhindern, daß das Programm *piq* zweimal nebeneinander gestartet wird. Es kann deshalb nach einem Absturz des Programmes notwendig werden, dieses File von Hand oder mit dem Shellskript *killpiq* zu löschen, um das Programm erneut starten zu können.

### piq_dump

ist das Protokollfile, in das die Protokollfunktion ihre Werte schreibt. Neue Werte werden grundsätzlich an das File angehängt, das File wird also nicht beim Programmstart gelöscht. Die Werte können nicht nachträglich eingelesen werden.

### piq_actual

Dieses File dient zur Online-Überwachung der Quelle und wird bei jedem Zeittakt neu geschrieben. Es wird z.B. vom Programm *piqwatch* gelesen.

### piq_panic

Bei einem Programmabsturz versucht das Programm, den aktuellen Quellenstatus in obigem Datenfile zu sichern, um einen Neustart zu ermöglichen. Dieses File hat dasselbe Format wie *piq_values* oder die durch die save-Funktion erzeugten Files und kann durch die Kommandozeilenoption -p zur Initialisierung benutzt werden.

### piq_alarm

Ist ein Shellskript, welches ausgeführt wird, wenn eine Alarmschwelle überschritten wird. In diesem Falle wird das File *piq_alarm_msg* mit einer entsprechenden Fehlermeldung geschrieben.

## A.13 Sonstiges

### Screen-Refresh

Ein Neuaufbau des Bildschirms kann durch Drücken der Taste 'r' oder durch den entsprechenden Button im X-Kommandofenster erreicht werden.

### load/save values

Diese Funktionen dienen zum Speichern/Laden des momentanen Quellenzustandes. Kommandozeilenbefehle: 'load ⟨file⟩' und 'save ⟨file⟩'. X-Fenster: Menüpunkte 'file/load' und 'file/save'.

Diese Funktionen haben nichts mit der Protokollierungsfunktion zu tun, da die bei der Protokollierung gespeicherten Werte nicht mehr eingelesen werden können.

## A.14 Fehlersuche

Im Falle einer Funktionsstörung der Fernsteuerung wird empfohlen, folgende Fehlerquellen vorrangig zu prüfen:

**Kompletter Ausfall des CAMAC-Systems:** Die Befestigung der CAMAC-Karte im PC sollte überprüft werden.

**Der SCC wird nicht initialisiert,** CAMAC-Kommandos lassen sich nicht an die Quelle schicken, die ADNR-LED des SHD leuchtet.

1. Der SHD erhält evt. zu wenig Strom. Er sollte versuchsweise in einem Crate ohne zusätzliche Bausteine betrieben werden.
2. Ansonsten liegt vermutlich ein Defekt des seriellen Kabels vor. Die Lötstellen der Stecker sollten auf Bruch oder Kurzschluß überprüft werden. Dasselbe gilt für die Turn-Round-Connectors.

**CAMAC-Kommandos kommen korrekt auf der Quelle an,** trotzdem leuchtet das ADNR-LED des SHD. Vermutlich sind die U-Port-Adapter falsch konfiguriert. Der Kodierstecker des AUX-Portes sollte überprüft und der Slave-UPA per Hand konfiguriert werden. Auch kann ein Defekt des seriellen Kabels vorliegen.

**Störung des Master-UPAs** bei Hochfahren des Potentials der Quellenplattform: Der UPA wird über das Stromnetz durch das Hochspannungsnetzgerät der Quellenplattform stark gestört. Er sollte an einem vom Netzanschluß des Netzgerätes entkoppelten Anschluß betrieben werden.

# Anhang B

# Technische Besonderheiten

## Polarisationsumschaltung

Der Schaltkasten für die Polarisationsumschaltung wies den Fehler auf, daß die Übernahme eines geschriebenen Wertes erst einen Schreibzyklus später erfolgte, d.h. für das Schreiben eines bestimmten Wertes mußte dieser Wert zweimal geschrieben werden. Dies lag an einer fehlerhaften Verdrahtung in der Schaltung, durch die das für die Umschaltung verantwortliche Flipflop die falsche Flanke des Schaltimpulses lieferte. Das Problem wurde durch Benutzung des nichtinvertierten statt wie ursprünglich des invertierten Ausganges gelöst.

## Fehler vom Kontrollpult

Die Erfahrung hat gezeigt, daß vom Kontrollpult ca. alle 2000 Übertragungen der Wert 0 anstelle des korrekten Wertes der Potentiometer bzw. der digitalen Schalter zum Kontrollpult übertragen wird. Die Ursache hierfür wird in induzierten Störungen vermutet, die eine Störung der Enable-Signale der Treiberplatinen verursachen. Als Folge legen diese nicht wie erwartet die korrekten Daten auf den Write-Bus, so daß sich beim Lesen dieses Busses der Wert 0 ergibt. Eine Verkleinerung der Pull-Up-Widerstände zur elektrischen Beseitigung der Störungen brachte keine Verbesserung.

Zur Vermeidung dieser Übertragungsfehler wird seitens der Software der Rückmeldewert 0 ignoriert. Bei den digitalen Schaltern ist dies völlig unkritisch, da diese immer einen Wert ungleich 0 besitzen (der Dir-Schalter befindet sich immer im Zustand Cup oder Tandem und erzeugt so ein Bitmuster $\neq 0$). Ebenfalls unkritisch ist dies bei den Potentiometern, da diese meistens einen Offset besitzen und so ebenfalls nicht den Wert 0 annehmen können. Bei Potentiometern ohne Offset treten auch keine Probleme auf, da die Werte der Potentiometer nur noch relativ interpretiert werden und der Wegfall eines einzelnen Wertes am Rand der Skala nicht von Bedeutung ist.

In intensiven Tests konnten weitere, andersartige Störungen nicht beobachtet werden. Insbesondere ergab die Prüfung der vom Kontrollpult gelesenen Werte, daß außer der 0 keine anderen unsinnigen Werte auftreten. Dies ist plausibel, da das IO440-CAMAC-Register die anliegenden Signale über einen gewissen Zeitraum integriert, so daß die Eingangsstufe dieses Registers erheblich unempfindlicher gegen Störungen ist als die Eingänge der Enable-Signale der Treiberplatinen, die direkt an TTL-ICs anliegen. Mit Hilfe der oben genannten Methode wird daher ein störungsfreier Betrieb des Kontrollpultes ermöglicht.

## Hardware-Fehler des SHD

Das RDY-Statusbit des SHD wird korrekt auf den LEDs der Frontplatte angezeigt und läßt sich durch die CAMAC-Funktion F27A1, die ein Q bei gesetztem Bit zurückliefert, korrekt testen. Beim Auslesen des Statusregisters mittels F1A12 wird dieses Bit jedoch häufig nicht korrekt übertragen. Dies ist mit großer Sicherheit auf einen Hardware-Defekt zurückzuführen. Das Problem wurde umgangen, indem das Bit im Steuerprogramm ausschließlich durch die Funktion F27A1 getestet wird.

## Bug im Betriebssystem

Das Steuerprogramm erhält beim ersten Start nach einem Systemboot nicht genügend Shared Memory. Bei jedem weiteren Startversuch startet das Programm problemlos. Der enge Zusammenhang mit dem Bootvorgang deutet stark auf einen Fehler im Betriebssystem[1] hin.

[1] Linux Kernel-Version 1.2.10 und 1.2.13

# Anhang C

# Tastaturbelegung

## C.1 Kommandozeilenoptionen piq

- **-s** : Programm übernimmt eingestellte Werte auf der Quelle als Startwerte.
- **-p** : Startwerte werden aus 'home/knecht3T/etc/piq_panic' gelesen.
- **-f** $< file >$ : Startwerte werden aus $< file >$ gelesen.
- **keine Parameter** : Startwerte werden aus '/home/fj/etc/piq_values' gelesen.

## C.2 analoge Parameter

| | |
|---|---|
| 0-9 | Auswahl des aktuellen Kanals (zweistellig) |
| +/− | Wert des aktuellen Kanals ±1 Skt |
| w/y | Wert des aktuellen Kanals ±10 Skt |
| e/x | Wert des aktuellen Kanals ±100 Skt |

## C.3 digitale Parameter

| | |
|---|---|
| u | Pol up |
| d | Pol down |
| n | Pol unpol |
| c | Cup |
| t | Tandem |
| a | Quench all on/off |

## C.4 Sonstiges

l/k/p Lock/Keyboard(=Terminal)/Pult-Modus

s Schreibt aktuelle Werte in Protokollfile

z (Zap Bit 16) Setzt/Löscht Extern-Anforderungs-Bit (für die Werteübernahme per Hand)

r Refresh screen

i Input Befehlsstring (siehe PIQ-Befehlsliste)

] Umschalten Kanal-Direktbelegung

## C.5 Direktbelegung

Je 2 Tasten entsprechen einem Kanal. Umschalten in Funktionsbelegung mit ].

| Nr. | +/- | Nr. | +/- | Nr. | +/- | Nr. | +/- |
|---|---|---|---|---|---|---|---|
| 1 | 1/q | 5 | 5/t | 9 | a/y | 13 | g/b |
| 2 | 2/w | 6 | 6/z | 10 | s/x | 14 | h/n |
| 3 | 3/e | 7 | 7/u | 11 | d/c | 15 | j/m |
| 4 | 4/r | 8 | 8/i | 12 | f/v | 16 | k/, |

# Anhang D

# Befehlsübersicht

Die folgenden Befehle können...

- ...als Kommandozeile des Befehls 'piqcmd' an das Steuerprogramm übergeben werden (z.B. durch einen rsh-Aufruf). Rückmeldung erfolgt in die Standardausgabe von 'piqcmd' . Der Befehl 'status' wird gesondert behandelt und gibt als Ergebnis den kompletten Quellenstatus zurück.
- ...über den tcl-Befehl 'piqsendcmd ⟨Befehl⟩ [Parameter...]' in ein Tcl/Tk-Programm eingebunden werden, welches mit der Shell 'piqwish' ausgeführt werden kann. Rückmeldung erfolgt in die Tcl-Variable 'PiqAnswer'
- ...direkt eingegeben werden durch Drücken der Taste 'i' im Xterm-Fenster des Piq-Programmes.
- ...im Setup-Skript 'piq_setup' ausgeführt werden. Siehe dazu besonders Punkt D.3.

## D.1 Analoge Parameter

**set ⟨channel⟩ ⟨float-value⟩** Setzt Kanal ⟨channel⟩ auf ⟨float-value⟩.

**iset ⟨channel⟩ ⟨int-value⟩** Setzt Kanal ⟨channel⟩ auf ⟨int-value⟩ Skalenteile.

**add ⟨channel⟩ ⟨int-value⟩** Addiert zu Kanal ⟨channel⟩ ⟨int-value⟩ Skalenteile.

## D.2 Digitale Parameter

**up/down/unpol** Setzt Polarisation.

**cup/tandem** Setzt Richtung des Strahls.

**qall/qoff** Setzt Quench-All-Status.

**quench** Invertiert Quench-All-Status.

**lockmode/termmode/pultmode** Ändert Eingabemodus.

**keymode** Synonym für termmode.

**potimode** Synonym für pultmode.

## D.3 Setup-Befehle

Diese Befehle sind speziell für die Verwendung im File 'piq_setup' gedacht.

**refresh_time** ⟨**sec**⟩ ⟨**msec**⟩ Setzt Zeitintervall des Refresh-Zyklus in Sekunden und Millisekunden. Ein sinnvoller Wert ist z.B. 500 msec.

**name** ⟨**channel**⟩ ⟨**name**⟩ Setzt den Namen des Kanals.

**unit** ⟨**channel**⟩ ⟨**unit**⟩ Setzt die Einheit des Kanals.

**runit** ⟨**channel**⟩ ⟨**unit**⟩ Setzt die Einheit des Read- (Ist-)Wertes des Kanals, falls diese von der Einheit des Sollwertes abweicht.

**scale_ra** ⟨**channel**⟩ ⟨**a**⟩ Setzen des Eichparameters a für den Read-Wert nach der Formel $y = b * x + a$ bei linearer und $y = a * exp(b * x)$ bei exponentieller Eichung.

**scale_rb** ⟨**channel**⟩ ⟨**b**⟩ Setzen des Eichparameters b für den Read-Wert.

**scale_wa** ⟨**channel**⟩ ⟨**a**⟩ Setzen des Eichparameters a für den Write-Wert.

**scale_wb** ⟨**channel**⟩ ⟨**b**⟩ Setzen des Eichparameters a für den Write-Wert.

**pos** ⟨**channel**⟩ ⟨**pos**⟩ Setzen der Zeilen-Position des Eintrages in der Bildschirmmaske.

**res** ⟨**channel**⟩ ⟨**res**⟩ Setzen der Auflösung des Potentiometers des Kanals. Die gelesenen Werte des Reglers werden um ⟨res⟩ Stellen nach rechts verschoben, so daß sich eine Untersetzung um Faktor $2^{\langle res \rangle}$ ergibt.

**min** ⟨**channel**⟩ ⟨**min**⟩ Setzen des Minimalwertes des Sollwertes eines Kanals in Skalenteilen. Bei Eingabe eines Wertes, der diesen Wert unterschreitet, bleibt der Sollwert auf diesem Wert stehen.

**max** ⟨**channel**⟩ ⟨**max**⟩ Setzen des Maximalwertes des Sollwertes eines Kanals in Skalenteilen. Bei Eingabe eines Wertes, der diesen Wert überschreitet, bleibt der Sollwert auf diesem Wert stehen.

**alarmx** ⟨**channel**⟩ ⟨**alarms**⟩ Setzen der oberen Alarmschwelle des Istwertes eines Kanals in Skt. Bei Überschreiten dieses Wertes wird das Alarmskript 'piq_alarm' ausgeführt.

**alarmn** ⟨**channel**⟩ ⟨**alarms**⟩ Setzen der unteren Alarmschwelle des Istwertes eines Kanals in Skt.

**alarmbit** ⟨**bit**⟩ ⟨**message**⟩ Definition eines Alarmbits. Wird im Eingabregister JRE0 der Quelle Bit Nr. ⟨bit⟩ gesetzt, wird die Meldung ⟨message⟩ auf dem Bildschirm ausgegeben.

**nalarmbit** ⟨**bit**⟩ ⟨**message**⟩ Entspricht der Inversion von alarmbit. Die Meldung wird ausgegeben, wenn das entsprechende Bit *nicht* gesetzt ist.

**flag** ⟨**channel**⟩ ⟨**flag**⟩ Setzen des Flags ⟨flag⟩ für Kanal ⟨channel⟩. Es können mehrere Flags für einen Kanal gesetzt werden, jedoch muß für jedes ein separater Flag-Befehl benutzt werden. Die Flags können folgende Werte haben:

- **no_extint:** Der Kanal besitzt keinen Extern/Intern Werteübernahmemechanismus.
- **no_return:** Der Kanal besitzt keine Rückmeldung.
- **no_send:** Der Kanal kann keinen Sollwert empfangen.
- **dead:** Der Kanal ist abgeschaltet.
- **i_exp_eich:**Die Eichung des Istwertes ist exponentiell statt linear.
- **no_follow:** Bei der automatischen Wertübernahme soll nicht versucht werden, den vorherigen Istwert durch Variation des Sollwertes wieder herzustellen.
- **poti_abs:** Das Potentiometer wird absolut statt relativ behandelt.

## D.4 Quellenstatus erfragen

**getdstat ⟨parameter⟩** Erfragt Status des digitalen Parameters ⟨parameter⟩, welcher 'pol', 'dir', 'quench' oder 'mode' sein kann. Rückmeldungen können sein:

- **pol :** up, down, unpol
- **dir :** cup, tandem
- **quench :** off, all
- **mode :** pult, terminal, lock

Im Fehlerfall wird 'error' zurückgegeben.

**getastat ⟨channel⟩ ⟨parameter⟩** Erfragt von Kanal ⟨channel⟩ den Status des analogen Parameters ⟨parameter⟩ , welches 'soll', 'ist', 'ext', 'name' oder 'unit' sein kann. Rückmeldungen können sein:

- **soll/ist :** digitaler Wert
- **ext :** Extern-Status: 0 oder 1
- **name :** Name des Kanals
- **unit :** Einheit des Kanals

Im Fehlerfall wird 'error' zurückgegeben.

**status** Der aktuelle Status wird als ASCII-Text im File '/home/fj/etc/piq_actual' ausgegeben.

## D.5 Laden/Speichern des Quellenstatus

**save ⟨file⟩** Speichert die aktuellen Werte der Quellenparameter im Datenfile ⟨file⟩.

**load ⟨file⟩** Setzt die Quellenparameter auf die in ⟨file⟩ gespeicherten Werte.

## D.6 Protokollfunktion

**dadd** ⟨**channel**⟩ ⟨channel⟩ wird zur Liste der protokollierten Kanäle hinzugefügt.

**dsub** ⟨**channel**⟩ ⟨channel⟩ wird aus Liste der protokollierten Kanäle entfernt.

**dshow** Zeigt Liste der protokollierten Kanäle.

**dstart** ⟨**time**⟩ Startet Protokoll mit Zeitintervall ⟨time⟩ sec.

**dend** Beendet Protokoll.

**dlist** ⟨**file**⟩ Liste der protokollierten Kanäle wird aus ⟨file⟩ eingelesen.

## D.7 Werteübernahme

**tkover** Automatische Übernahme der Quellenwerte. Dieser Befehl sollte nur direkt am Terminal eingegeben werden und nicht z.B. aus einem Tcl/Tk-Programm aufgerufen werden, da er eine Bestätigung im Terminalfenster verlangt.

**getpoti** Wert aller Potentiometer wird als Absolutwert übernommen.

**getspoti** ⟨**nr**⟩ Wert des Potentiometers Nr. ⟨nr⟩ wird als Absolutwert übernommen.

**bit16** ⟨**0/1**⟩ Setzt Extern-Anforderungs-Bit auf 0 oder 1.

## D.8 Sonstiges

**bl2down** Fährt BL2 auf 0.

**help** Gibt Online-Hilfe aus.

**rscreen** Baut Bildschirm neu auf.

**setac** ⟨**channel**⟩ Aktueller Kanal wird auf ⟨channel⟩ gesetzt.

**clientid** ⟨**PID**⟩ Setzt Client-ID, an die die Antwort erfolgen soll, auf ⟨PID⟩.

**resalarm** Setzt Alarmflag zurück. Dieses wird gesetzt, falls eine Alarmschwelle überschritten wurde und das Alarmskript ausgeführt wurde. Damit soll verhindert werden, daß es zur dauernden Ausführung des Skriptes kommt. Ist die Ursache des Alarms beseitigt, muß dieses Flag durch obigen Befehl zurückgesetzt werden.

**cam** ⟨$B$⟩⟨$C$⟩⟨$N$⟩⟨$A$⟩⟨$F$⟩⟨$V$⟩ Sendet CAMAC-Kommando. Branch 7 entspricht serieller Übertragung.

**loopc** Setzt den Loop-Collapse-Status des SCC.

**sccinit** Sendet Initialisierungskommando an den SCC.

**quit** Beendet Programm.

# Anhang E

# Tabellen

Tabelle E.1: Interne Zusatzverkabelung in der Aniol-Zentrale

| Signal | Bedeutung | Pin Platine | Pin Stecker | Kabel |
|---|---|---|---|---|
| M0 | Select BT1 | 4 BT1 | R 34 | ws |
| M1-1 | Select BT3 | 3 BT3 | R 35 | br |
| Z1 | Strobe DAS | 3 DAS | R 36 | gn |
| Z3 | Strobe AAV | 3 AAV | R 37 | gl |
| Z4 | Strobe AAV | 8 AAV | R 38 | gr |
| Z5 | Strobe Ext/Int-LEDs | 30 FBE-Stecker | R 46 | rt |
| ADC S.C. | Strobe ADC | 15 AN87 | R 39 | rs |
| EOC | End of. Conv. | 3 P2 | L 23 | gl/ws |
| A | Kanal MUX | 13 AN87 | L 18 | bl |
| B | Kanal MUX | L AN87 | L 19 | rt |
| C | Kanal MUX | K AN87 | L 20 | sw |
| D | Kanal MUX | 10 AN87 | L 21 | vi |
| E | (Kanal MUX) | NC | L 22 | gn/ws |

Tabelle E.2: Belegung IO440-Out / Aniol-In

| Signal | Aniol R | Kabel | IO440 Out | Bedeutung |
|---|---|---|---|---|
| R1 | 1 | gl/bl | 1 | Write-Bus |
| R2 | 2 | bl/sw | 2 | ” |
| R3 | 3 | ws/bl | 3 | ” |
| R4 | 4 | bl | 4 | ” |
| R5 | 5 | or/bl | 5 | ” |
| R6 | 6 | or | 6 | ” |
| R7 | 7 | rt | 7 | ” |
| R8 | 8 | gn | 8 | ” |
| R9 | 9 | rt/br | 9 | ” |
| R10 | 10 | rs | 10 | ” |
| R11 | 11 | tk | 11 | ” |
| R12 | 12 | gl/gn | 12 | ” |
| R13 | 13 | sw | 13 | ” |
| R14 | 14 | ws/gn | 14 | ” |
| R15 | 15 | rt/sw | 15 | ” |
| R16 | 16 | br | 16 | ” |
| M0 | 34 | vi | 17 | Enable BT1 (ADC-Daten) |
| M1-1 | 35 | or/gn | 18 | Enable BT3 (Digit.-Schalter) |
| Z1 | 36 | ws/rt | 19 | Strobe DAS (Digit.-Anz.) |
| Z3 | 37 | gl/rt | 20 | Strobe AAV Latch (An.-Anz.) |
| Z4 | 38 | rt/bl | 21 | Strobe AAV 16-aus-4 |
| Z5 | 46 | gl | 23 | Strobe Ext/Int LEDs FBE |
| R12+R13 | 29 | ” | ” | ” |
| ADC SC | 39 | ws | 22 | ADC Start Conv. |
| NC | 47 | gr/bl | 24 | Reserve |
| GND | 49 | screen | 25 | Masse |

Tabelle E.3: Belegung IO440-In / Aniol-Out

| Signal | Aniol L | Kabel | IO440 In | Bedeutung |
|---|---|---|---|---|
| W1 | 1 | rs | 1 | Read-Bus |
| W2 | 2 | gn | 2 | ” |
| W3 | 3 | or/gr | 3 | ” |
| W4 | 4 | gl/bl | 4 | ” |
| W5 | 5 | gl/gn | 5 | ” |
| W6 | 6 | bl/sw | 6 | ” |
| W7 | 7 | sw | 7 | ” |
| W8 | 8 | rt | 8 | ” |
| W9 | 9 | br | 9 | ” |
| W10 | 10 | rt/br | 10 | ” |
| W11 | 11 | ws/gn | 11 | ” |
| W12 | 12 | ws/bl | 12 | ” |
| W13 | 13 | tk | 13 | ” |
| W14 | 14 | bl | 14 | ” |
| W15 | 15 | ws | 15 | ” |
| W16 | 16 | or/bl | 16 | ” |
| EOC | 23 | gr | 17 | End of Conversion ADC |
| A | 18 | gl | 18 | Bit 1 des aktuellen ADC-Kanals |
| B | 19 | vi | 19 | Bit 2 ” |
| C | 20 | ws/rt | 20 | Bit 3 ” |
| D | 21 | rt/sw | 21 | Bit 4 ” |
| E | 22 | or | 22 | (nicht benutzt) |
| NC | - | gn/br | 23 | - |
| NC | - | gl/rt | 24 | - |
| GND | 49 | screen | 25 | Masse |

Tabelle E.4: Serielles Kabel

| Pin | Signal | Kabel |
|---|---|---|
| 20/21 | Bus 2 (Clk ret) | gn |
| 22/23 | Clk | ws |
| 2/3 | Bus 1 (Bit ret) | bl |
| 4/5 | Bit | rt |

Tabelle E.5: AUX-Connector

| Signal | Pin | Schalter |
|---|---|---|
| 0A | 23 | 1 |
| 1A | 21 | 2 |
| 2A | 9 | 3 |
| 0B | 10 | 4 |
| 1B | 8 | 5 |
| 2B | 22 | 6 |

Tabelle E.6: Turn-Round-Connector

| von Pin | auf Pin |
|---|---|
| 2 | 4 |
| 3 | 5 |
| 20 | 22 |
| 21 | 3 |

Tabelle E.7: Read-Werte (Digital-Schalter)

| Write-Leitungen | W16 | W8/W7 | W6/W5 | W4/W3 | W2/W1 |
|---|---|---|---|---|---|
| Bit Nr. JRS0 | 15 | 7/8 | 4/5 | 2/3 | 0/1 |
| Bedeutung | Quench all | Cup | Tandem | Down | Up |

Tabelle E.8: Write-Werte (Ansteuerung LEDs)

| Read-Leitungen | R13 | R5 | R4 | R3 | R2 | R1 |
|---|---|---|---|---|---|---|
| Bit Nr. JRE0 | 12 | 4 | 3 | 2 | 1 | 0 |
| Bedeutung | Turbos Ok | Tandem | Cup | Unpol | Down | Up |

# Literaturverzeichnis


[An76] Roland Aniol, Entwicklung und Aufbau der Fernsteuerung für die Polarisierte-Ionen-Quelle am Kölner Tandem-van-de-Graaf-Beschleuniger, Diplomarbeit, Institut f. Kernphysik der Univ. Köln, 1976

[Be78] V. Bechtold, L. Friedrich, P. Ziegler, R. Aniol, G. Latzel, H. Paetz gen. Schieck, Nucl. Inst. Meth 150 (1978) 407

[Be94] M. Beck et al., Linux-Kernel-Programmierung, Addison Wesley 1994

[Eg94] Markus Eggert, Entwicklung eines Kontrollsystems für die neue Quelle polarisierter Ionen bei COSY/Jülich, Diplomarbeit, Institut f. Kernphysik der Univ. Köln, 1994

[Fr95] Claudia Franke, Aufbau und Test eines Lambshift-Spinfilters, Diplomarbeit, Institut f. Kernphysik der Univ. Köln, 1995

[He95] S. Hetze, D. Hohndel, O. Kirch, M. Müller, LinuX Anwenderhandbuch, LunetIX Softfair 1995

[Hy01] CAMAC Serial Highway Driver SHD 992 User Manual, Hytec Electronics Ltd., 64 Amy Street, Leicester, LE3 2FB England, c/o Fa. Struck, Postfach 1141, D-22886 Tangstedt

[Hy02] CAMAC Serial Crate Controller SCC2401, Hytec Electronics

[Hy03] HYTEC Manual for U-Port Adaptors, Hytec Electronics

[Hy04] Hytec 440-1-4 IO-Register, Hytec Electronics

[Ie71] CAMAC Tutorial Issue, IEEE Transactions on Nuclear Science, No. 2 April 1971

[Ke88] B.W. Kernighan, D.M. Ritchie, The C Programming Language, Prentice Hall 1988

[Kr96] Gerhard Krause, Optimierung der Steuerung und der Datenaufnahme an der Quelle polarisierter Ionen COSY/Jülich, Diplomarbeit, Institut für Kernphysik der Univ. Köln, vorrauss. Mai 96.

[Le79] A. Lesea, R. Zaks, Microprocessor Interface-Techniken,Sybex 1979

[Li01] Linux Documentation Project im World Wide Web, http://sunsite.unc.edu/mdw/-linux.html

[Mü94] Uwe Müssel, private Mitteilung, Institut für Kernphysik der Univ. Köln 1994

[Ni94] Norbert Nicolay, CAMAC-Library für LynxOS, private Mitteilung, Institut für Kernphysik der Univ. Köln 1994

[Ou94] John K. Ousterhout, Tcl and the Tk Toolkit, Addison Wesley 1994

[Re89] Ralf Reckenfelderbäumer, Untersuchungen an der Kölner Lambshift-Quelle für polarisierte Ionen, Diplomarbeit, Institut f. Kernphysik der Univ. Köln, 1989

[St92] W. Richard Stevens, Advanced Programming in the UNIX Environment, Addison Wesley 1992

[Tr94] Alexandra Trützschler, Entwicklung eines Strahltargetpolarimeters für das EDDA-Experiment an COSY, Diplomarbeit, Institut f. Kernphysik der Univ. Köln, 1994

[We95] Brent Welch, Practical Programming in Tcl and Tk , Prentice Hall 1995

[Wi01] CAMAC Crate Controller CC16, Wiener Plein & Baus GmbH, Müllersbaum 20, D-51399 Burscheid

# Danksagung


- Herrn Professor Dr. H. Paetz gen. Schieck möchte ich für die Erteilung der vorliegenden Arbeit und die Unterstützung bei der Durchführung danken.
- Ich danke allen Mitgliedern der Arbeitsgruppe für die gute Zusammenarbeit, viele hilfreiche Ratschläge und tatkräftige Unterstützung, insbesondere Markus Eggert, Gerhard Krause, Lutz Sydow und Sascha Vohl.
- Mein besonderer Dank gilt Ralf Reckenfelderbäumer für die gute Betreuung dieser Arbeit.
- Norbert Nicolay danke ich für die Bereitstellung der CAMAC-Library unter LynxOS.
- Den Mitarbeitern der E-Werkstatt und den Tandemoperateuren möchte ich für die direkte Hilfe bei allen technischen Problemen danken.

## Erklärung

Ich versichere, daß ich die von mir vorgelegte Diplomarbeit selbständig und ohne unzulässige Hilfsmittel angefertigt und die benutzten Quellen und Zitate vollständig angegeben habe.

Köln, den 6. März 1996